CS3165M Software Engineering

客观题题库 (v3.0)

memset0 memset0@outlook.com

本题目集由 @memset0 制作，如有问题可扫描右侧二维码在 CC98 留言反馈.

- 题面爬取自课程组官网，感谢 @小角龙 学长.
- 使用 GPT 4.1 生成题目的中文翻译与解析.
- 因原网站只支持单选，多选题通过增设 E 选项给出，现将其拆分为期末考形式的多选题.

( ) **1-1. Which question no longer concerns the modern software engineer?** 1

A. Why does computer hardware cost so much?
B. Why does software take a long time to finish?
C. Why does it cost so much to develop a piece of software?
D. Why can't software errors be removed from products prior to delivery?

选项A“为什么计算机硬件如此昂贵？”已不是现代软件工程师主要关心的问题. 随着技术进步，硬件价格大幅下降，硬件成本已不再是软件开发的主要障碍. B、C、D涉及开发周期长、成本高、缺陷难以完全消除，这些仍是软件工程中的核心挑战.

(A) 1. 哪个问题已经不再困扰现代软件工程师?
A. **为什么计算机硬件这么贵?**
B. 为什么软件开发需要这么长时间?
C. 为什么开发一款软件的成本这么高?
D. 为什么在交付前无法彻底消除软件错误?

( ) **1-2. Software is a product and can be manufactured using the same technologies used for other engineering artifacts** 2

软件虽然是一种产品，但与传统工程产品(如汽车、机械)不同. 软件开发不能完全依赖传统的制造技术和工艺，因为软件是无形的，不涉及物理制造过程. 软件开发更侧重于设计、编码、测试和维护，强调思维和逻辑，不像传统工程那样通过装配线大规模生产. 因此，不能用制造其他工程产品的技术来制造软件.

(F) 2. 软件是一种产品，可以使用与其他工程制品相同的技术进行制造.

( ) **1-3. Software deteriorates rather than wears out because** 3

A. Software suffers from exposure to hostile environments
B. Defects are more likely to arise after software has been used often
C. Multiple change requests introduce errors in component interactions
D. Software spare parts become harder to order

软件不会因物理磨损损坏，而是在持续修改和维护中，反复的变更请求(如增加功能、修复缺陷)容易引入新错误，特别是在各组件交互时出现新问题，这就是“软件老化”或“软件蜕变”. 多次变更导致组件交互错误，是软件变质的主要原因. 其他选项要么描述不准确，要么并不适用于软件.

(C) 3. 软件会退化而不是磨损的原因是
A. 软件会因暴露在恶劣环境中而受损
B. 软件在被频繁使用后更容易出现缺陷
C. **多次变更请求会在组件交互中引入错误**
D. 软件的备件变得更难订购

( ) **1-4. WebApps are a mixture of print publishing and software development, making their development outside the realm of software engineering practice.** 4

WebApps(网络应用)虽然结合了内容呈现和用户体验设计，但本质上仍是软件系统. 它们的开发过程适用于软件工程的方法和规范，如需求分析、设计、编码、测试和维护等，因此WebApps的开发属于软件工程的实践范畴.

(F) 4. Web应用程序是印刷出版和软件开发的混合体，因此它们的开发不属于软件工程实践的范畴.

( ) **1-5. There are no real differences between creating WebApps and MobileApps.** 5

Web应用和移动应用在开发过程有许多区别，如技术栈不同、运行平台不同、用户体验要求不同、性能优化方式和交互设计也有差异. 因此，开发两者时需要考虑的方面和方法有明显区别.

(F) 5. 创建Web应用程序和移动应用程序之间没有真正的区别.

( ) **1-6. In its simplest form an external computing device may access cloud data services using a web browser.** 6

最简单时，外部计算设备(如电脑、平板或手机)可以用网页浏览器访问云数据服务. 大多数云服务(如云存储、云办公等)都提供网页界面，用户只需用浏览器打开相关网址，就能访问和操作云端数据，无需额外安装软件.

(T) 6. 在最简单的形式下，外部计算设备可以使用网页浏览器访问云数据服务.

( ) **1-7. Product line software development depends the reuse of existing software components to provide software engineering leverage.** 7

产品线软件开发(Product Line Software Development)的核心是重用已有的软件组件，用于快速开发不同但相关的软件产品. 这能提高开发效率，降低成本，并保证产品的一致性和质量. 因此，产品线开发依赖已有软件组件的复用.

(T) 7. 产品线软件开发依赖于现有软件组件的复用，以提供软件工程的杠杆作用.

( ) **2-1. Which of the items listed below is not one of the software engineering layers?** 8

A. Process
B. Manufacturing
C. Methods
D. Tools

软件工程有四个主要层次：过程(Process)、方法(Methods)、工具(Tools)和质量关注(Quality Focus). “过程”用于管理和组织开发，“方法”指开发和分析软件的技术与步骤，“工具”是支持过程和方法的自动化工具. “制造(Manufacturing)”属于传统工业生产，不属于软件工程层次结构.

(B) 1. 下列哪一项不是软件工程的层次?
A. 过程
B. **制造**
C. 方法
D. 工具

( ) **2-2. Software engineering umbrella activities are only applied during the initial phases of software development projects.** 9

软件工程中的伞形活动(umbrella activities)，如项目管理、质量保证、配置管理等，贯穿整个软件开发生命周期，不只在初始阶段. 这些活动会在项目的开始、进行和结束阶段持续进行，以保证开发顺利和高质量完成. 因此，“只在初始阶段应用”是不对的.

(F) 2. 软件工程的支撑活动只在软件开发项目的初始阶段应用.

( ) **2-3. Which of these are the 5 generic software engineering framework activities?** 10

A. communication, planning, modeling, construction, deployment
B. communication, risk management, measurement, production, reviewing
C. analysis, designing, programming, debugging, maintenance
D. analysis, planning, designing, programming, testing

软件工程的五大通用框架活动包括communication(沟通)、planning(计划)、modeling(建模)、construction(构建)、deployment(部署). 这五个活动几乎贯穿所有软件开发过程，是软件工程的核心流程. 其他选项虽涉及开发环节，但未覆盖所有框架活动或顺序不正确.

(A) 3. 以下哪些是5个通用的软件工程框架活动?
A. **沟通、计划、建模、构建、部署**
B. 沟通、风险管理、度量、生产、评审
C. 分析、设计、编程、调试、维护
D. 分析、计划、设计、编程、测试

( ) **2-4. Planning ahead for software reuse reduces the cost and increases the value of the systems into which they are incorporated.** 11

提前为软件复用做规划可以让软件组件高效重复使用，减少重复开发，降低开发和维护成本. 复用的软件通常质量较高，经过多次验证，提升了系统的稳定性和可维护性，从而整体提升系统价值. 因此，软件工程强调在项目早期就考虑复用.

(T) 4. 提前规划软件复用可以降低成本，并提高所集成系统的价值.

( ) **2-5. The essence of software engineering practice might be described as understand the problem, plan a solution, carry out the plan, and examine the result for accuracy.** 12

软件工程实践的本质包括：理解问题(需求分析)、规划解决方案(设计)、执行计划(实现)、检查结果是否准确(测试和评审). 这是软件工程的基本流程，也是解决复杂问题常用的方法论.

(T) 5. 软件工程实践的本质可以描述为：理解问题、规划解决方案、执行计划，并检查结果的准确性.

( ) **2-6. In agile process models the only deliverable work product is the working program.** 13

敏捷过程模型虽然强调“可工作的软件”，但实际开发中还会有其他可交付成果，如用户故事、测试用例、文档、设计草图等. 这些内容在敏捷开发中同样重要，不只是程序才是唯一的交付物.

(F) 6. 在敏捷过程模型中，唯一的可交付工作产品是可运行的程序.

( ) **2-7. A most software development projects are initiated to try to meet some business need.** 14

绝大多数软件开发项目的启动，都是为了满足某种业务需求. 企业或组织通过开发软件提升效率、增加收入、改善服务或解决特定问题. 软件工程项目通常围绕实际的业务目标展开，而不是单纯为了技术本身.

(T) 7. 大多数软件开发项目的启动是为了满足某些业务需求.

( ) **2-8. In general software only succeeds if its behavior is consistent with the objectives of its designers.** 15

软件能否成功，关键在于是否满足用户或客户的需求和期望，而不仅仅是设计者的目标. 即使软件完全按照设计者的想法实现，如果没有解决用户实际的问题，软件仍然会失败. 软件成功的标准应是其行为是否和用户目标一致，而不是设计者的目标.

(F) 8. 一般来说，只有当软件的行为与其设计者的目标一致时，软件才会成功.

( ) **3-1. Which of the following are recognized process flow types?** 16

A. Concurrent process flow
B. Iterative process flow
C. Linear process flow
D. Spiral process flow

本题考查软件工程中的过程流(process flow)类型．经典的软件开发生命周期常见的过程流有：Linear process flow(线性过程流)，如瀑布模型，开发过程按顺序进行；Concurrent process flow(并发过程流)，多个开发活动可以同时进行．而Iterative process flow(迭代过程流)和Spiral process flow(螺旋过程流)严格来说属于开发模型(process model)或生命周期模型，不是过程流类型．

(AC) 1. 下列哪些是公认的过程流程类型?
A. **并发过程流程**
B. 迭代过程流程
C. **线性过程流程**
D. 螺旋过程流程

( ) 3-2. The communication activity is best handled for small projects using six distinct actions (inception, elicitation, elaboration, negotiation, specification, validation). 17

“六个明确的活动”——inception(开始)、elicitation(获取)、elaboration(细化)、negotiation(协商)、specification(规格说明)、validation(验证)——是需求分析阶段常见的活动，但并不适用于所有小型项目．小型项目的沟通活动通常可以更简化，不必分为这么多步骤，有时几个步骤会合并进行，过程更灵活．因此，“best handled for small projects using six distinct actions”这个说法不准确．

(F) 2. 对于小型项目，沟通活动最好通过六个不同的动作来处理(启动、引导、细化、协商、规范、验证)．

( ) 3-3. A good software development team always uses the same task set for every project to insure high quality work products 18

题目考查软件开发团队在不同项目中任务集(task set)的灵活性．不同项目在规模、复杂性、需求和目标上各不相同，如果每个项目都用相同的任务集，流程会变得僵化，无法满足具体项目需求，甚至影响产品质量．优秀的软件开发团队会根据项目实际情况调整和定制任务集，以确保高质量的工作成果．

(F) 3. 一个优秀的软件开发团队总是为每个项目使用相同的任务集，以确保高质量的工作产品．

( ) 3-4. Software processes can be constructed out of pre-existing software patterns to best meet the needs of a software project. 19

软件过程可以通过组合已有的软件模式(如设计模式、开发流程模式等)来构建，从而更好地满足具体项目需求．这样可以利用已有的最佳实践，提高开发效率和软件质量，也能根据项目特点灵活调整过程．

(T) 4. 软件过程可以由预先存在的软件模式构建，以最佳满足软件项目的需求．

( ) 3-5. Which of these are standards for assessing software processes? 20

A. SEI
B. SPICE
C. ISO 9000
D. ISO 9001

SEI是一个机构，不是标准．SPICE(软件过程改进与能力评定，ISO/IEC 15504)和ISO 9001都是用于评估和改进软件过程的国际标准．ISO 9000是质量管理体系标准族，范围较广，不专门针对软件过程，而ISO 9001是规定质量管理体系要求的标准，更适合实际评估．

(BD) 5. 以下哪些是用于评估软件过程的标准?
A. SEI
B. **SPICE**
C. ISO 9000
D. **ISO 9001**

( ) 4-1. The waterfall model of software development is 21

A. A reasonable approach when requirements are well defined.
B. A good approach when a working program is required quickly.
C. The best approach to use for projects with large development teams.
D. An old fashioned model that is rarely used any more.

瀑布模型强调需求明确、开发流程线性推进，适用于需求清楚、变更少的情况．B描述的是快速原型等模型，C不一定适合大团队，D说法不准确，瀑布模型在某些场景下仍然使用．

(A) 1. 瀑布模型的软件开发是
A. **当需求定义明确时的一种合理方法．**
B. 当需要快速获得可运行程序时的一种好方法．
C. 适用于拥有大型开发团队项目的最佳方法．
D. 一种过时且很少再被使用的模型．

( ) 4-2. The incremental model of software development is 22

A. A reasonable approach when requirements are well defined.
B. A good approach when a working core product is required quickly.
C. The best approach to use for projects with large development teams.
D. A revolutionary model that is not used for commercial products.

增量模型将软件分成多个小部分(增量)逐步开发和交付．它可以先快速实现一个可运行的核心产品，在后续迭代中完善和添加新功能．当需要快速得到可用产品原型并逐步改进时，增量模型很适用，这与选项B描述一致．其他选项不是对增量模型的准确描述或与其不符．

(B) 2. 软件开发的增量模型是
A. 当需求被明确定义时的一种合理方法．
B. **当需要快速获得可运行的核心产品时的一种良好方法．**
C. 适用于拥有大型开发团队项目的最佳方法．
D. 一种革命性的模型，但不用于商业产品．

( ) 4-3. Evolutionary software process models 23

A. Are iterative in nature.
B. Can easily accommodate product requirements changes.
C. Do not generally produce throwaway systems.
D. All of the above.

进化型软件过程模型(Evolutionary software process models)具有迭代性(A)，通过重复开发和完善逐步构建系统；能较好地适应需求变化(B)，每次迭代都可根据新需求调整；通常不会产生一次性抛弃的系统(C)，每一阶段的产物都是最终系统的一部分．

(D) 3. 演化型软件过程模型
A. 具有迭代性．
B. 能够轻松适应产品需求的变化．
C. 通常不会产生一次性系统．
D. **以上都是．**

( ) 4-4. The prototyping model of software development is 24

A. A reasonable approach when requirements are well defined.
B. A useful approach when a customer cannot define requirements clearly.
C. The best approach to use for projects with large development teams.
D. A risky model that rarely produces a meaningful product.

原型模型适用于客户需求不明确的情况，通过快速构建原型，让客户直观看到系统雏形，提出需求和改进意见，开发团队根据反馈完善系统．选项B描述了原型模型的适用场景，其他选项与原型模型特点不符或说法错误．

(B) 4. 软件开发的原型模型是
A. 当需求定义清楚时的一种合理方法．
B. **当客户无法清晰定义需求时的一种有用方法．**
C. 适用于拥有大型开发团队项目的最佳方法．
D. 一种很少产生有意义产品的高风险模型．

( ) 4-5. The spiral model of software development 25

A. Ends with the delivery of the software product.
B. Is more chaotic than the incremental model.
C. Includes project risks evaluation during each iteration.
D. All of the above.

螺旋模型的核心是在每次迭代中进行风险分析和评估，这是区别于其他软件开发模型的最大特点．它并不比增量模型更混乱，也不是仅在交付产品时结束，而是在每个开发周期都关注风险．

(C) 5. 软件开发的螺旋模型
A. 以交付软件产品为结束．
B. 比增量模型更加混乱．
C. **在每一次迭代中都包含项目风险评估．**
D. 以上都是．

( ) 4-6. The concurrent development model is 26

A. Another name for concurrent engineering.
B. Defines events that trigger engineering activity state transitions.
C. Only used for development of parallel or distributed systems.
D. Used whenever a large number of change requests are anticipated.

并发开发模型(concurrent development model)有时被称为并发工程(concurrent engineering)．它用状态图描述开发活动(如分析、设计、实现等)之间的状态及其转换，这些转换由特定事件触发．并发开发模型不限于并行或分布式系统开发，适用于更广泛的软件开发过程．它也不是专门为大量变更请求而设计，虽然能处理变更，但这不是其核心．

(AB) 6. 并发开发模型是
A. **并发工程的另一种称呼．**
B. **定义触发工程活动状态转换的事件．**
C. 仅用于并行或分布式系统的开发．
D. 在预期有大量变更请求时使用．

( ) 4-7. The component-based development model is 27

A. Only appropriate for computer hardware design.
B. Not able to support the development of reusable components.
C. Dependent on object technologies for support.
D. Not cost effective by known quantifiable software metrics.

组件化开发模型(component-based development model)依赖于面向对象技术(object technologies)支持．面向对象技术提供封装、继承、多态等机制，使功能模块能被封装为独立可复用组件，通过标准接口进行集成和复用，这是组件化开发的核心．A项错误，组件化开发不仅用于硬件设计，更广泛应用于软件开发．B项错误，组件化开发的目标就是支持可复用组件的开发．D项错误，组件化开发通常提高开发效率和质量，被认为具有成本效益．

(C) 7. 基于构件的软件开发模型是
A. 只适用于计算机硬件设计．
B. 无法支持可复用构件的开发．
C. **依赖于面向对象技术的支持．**
D. 根据已知的可量化软件度量标准来看并不具有成本效益．

( ) 4-8. The formal methods model of software development makes use of mathematical methods to 28

A. Define the specification for computer-based systems.
B. Develop defect free computer-based systems.

C. Verify the correctness of computer-based systems.
D. All of the above.

形式化方法模型在软件开发中通过数学方法，精确定义系统需求(A)，开发高可靠性、缺陷极少甚至无缺陷的系统(B)，并验证系统的正确性(C)。A、B、C都属于形式化方法模型的应用.

(D) 8. 软件开发的形式化方法模型利用数学方法来
A. 定义基于计算机系统的规格说明.
B. 开发无缺陷的基于计算机系统.
C. 验证基于计算机系统的正确性.
D. **以上所有.**

( ) 4-9. Which of these is not one of the phase names defined by the Unified Process model for software development? 29

A. Inception phase
B. Elaboration phase
C. Construction phase
D. Validation phase

统一过程(Unified Process, UP)模型将软件开发分为四个阶段：Inception(初始阶段)、Elaboration(细化阶段)、Construction(构建阶段)、Transition(移交阶段)。Validation phase(验证阶段)不是统一过程模型正式定义的阶段名称.

(D) 9. 以下哪一项不是统一过程模型中定义的软件开发阶段名称?
A. 初始阶段
B. 细化阶段
C. 构建阶段
D. **验证阶段**

( ) 4-10. Which of these is not a characteristic of Personal Software Process? 30

A. Emphasizes personal measurement of work product.
B. Practitioner requires careful supervision by the project manager.
C. Individual practitioner is responsible for estimating and scheduling.
D. Practitioner is empowered to control quality of software work products.

Personal Software Process(个人软件过程，PSP)强调开发人员个人的自我管理和过程改进，核心是开发人员对自己的工作负责，包括工作量测量、进度估算与安排、以及对工作质量的控制。选项B错误，因为PSP强调个人自律和自主性，不需要项目经理的密切监督。选项A、C、D都是PSP的典型特点.

(B) 10. 以下哪一项不是个人软件过程(Personal Software Process, PSP)的特征?
A. 强调对个人工作产品的度量.
B. **实践者需要项目经理的严格监督.**
C. 个人实践者负责估算和进度安排.
D. 实践者有权控制软件工作产品的质量.

( ) 4-11. Which of these are objectives of Team Software Process? 31

A. Accelerate software process improvement
B. Allow better time management by highly trained professionals
C. Build self-directed software teams
D. Show managers how to reduce costs and sustain quality

Team Software Process(TSP，团队软件过程)的核心目标是帮助软件开发团队提升自我管理和自我组织能力，使成员能够独立、高效协作完成项目，所以C"打造自我管理的软件团队"属于其主要目标。TSP也强调让专业人员更好地管理自己的时间和工作计划，因此B"让高素质的专业人员更好地进行时间管理"也是其目标。A"加速软件过程改进"和D"向管理者展示如何降低成本并保持质量"不是TSP的直接目标.

(BC) 11. 以下哪些是团队软件过程(Team Software Process, TSP)的目标?
A. 加速软件过程改进
B. **让高素质的专业人员更好地进行时间管理**
C. **构建自我管理的软件团队**
D. 向管理者展示如何降低成本并保持质量

( ) 4-12. Process technology tools allow software organizations to compress schedules by skipping unimportant activities. 32

过程技术工具用于改进和规范开发流程，提高效率和质量，但并不是通过跳过某些开发流程活动来压缩进度。每个活动都有其价值，随意跳过会带来风险。正确做法是优化和自动化流程，而不是省略关键步骤.

(F) 12. 过程技术工具允许软件组织通过跳过不重要的活动来压缩进度.

( ) 4-13. It is generally accepted that one cannot have weak software processes and create high quality end products. 33

一般认为，弱的软件过程无法产生高质量的最终产品。软件过程是开发软件时遵循的方法、步骤和规范。如果过程弱，缺乏规范、标准和控制，容易导致需求不清、设计缺陷、测试不充分等问题，影响软件质量。因此，强有力的软件过程对保证最终产品的高质量非常关键.

(T) 13. 人们普遍认为，弱的软件过程无法产生高质量的最终产品.

( ) 5-1. Agility is nothing more than the ability of a project team to respond rapidly to change. 34

敏捷(Agility)不仅仅是项目团队对变化做出快速响应的能力，还包括持续与客户沟通、不断交付有价值的软件、自我管理团队、技术卓越等方面。敏捷是一整套价值观和原则，而不仅仅是"快速反应".

(F) 1. 敏捷性仅仅是指项目团队对变化做出快速响应的能力.

( ) 5-2. Which of the following is not necessary to apply agility to a software process? 35

A. Eliminate the use of project planning and testing
B. Only essential work products are produced
C. Process allows team to streamline tasks
D. Uses incremental product delivery strategy

选项A"消除项目计划和测试"并不是实现敏捷开发所必需的。敏捷方法强调灵活和快速响应变化，但并不意味着完全不做计划和测试，敏捷开发同样重视适度的计划和持续的测试。B、C、D选项体现了敏捷开发的重要特征：只产生必要的工作产品、流程允许团队简化任务、采用增量产品交付策略。因此A不是应用敏捷所必需的.

(A) 2. 下列哪一项在将敏捷性应用于软件过程时不是必需的?
A. **消除项目规划和测试的使用**
B. 只产出必要的工作产品
C. 过程允许团队简化任务
D. 采用增量式产品交付策略

( ) 5-3. How do you create agile processes to manage unpredictability? 36

A. Requirements gathering must be conducted very carefully
B. Risk analysis must be conducted before planning takes place
C. Software increments must be delivered in short time periods
D. Software processes must adapt to changes incrementally

敏捷开发(agile processes)核心是快速响应变化和持续交付可用的软件。C选项强调小步快跑和频繁迭代，D选项强调对变化的响应和过程的灵活调整，这两点都是敏捷开发的重要原则。A和B虽然有意义，但不是敏捷应对不确定性的关键方法.

(CD) 3. 如何创建敏捷流程以管理不可预知性?
A. 需求收集必须非常仔细地进行
B. 风险分析必须在计划之前进行
C. **软件增量必须在较短的时间周期内交付**
D. **软件过程必须逐步适应变化**

( ) 5-4. In agile software processes the highest priorities is to satisfy the customer through early and continuous delivery of valuable software. 37

敏捷方法强调最高优先级是通过早期并持续交付有价值的软件来满足客户需求，这体现在敏捷宣言的第一条："我们最重要的目标是通过及早和持续交付有价值的软件来满足客户。"

(T) 4. 在敏捷软件开发过程中，最高优先级是通过及早和持续交付有价值的软件来满足客户.

( ) 5-5. In agile development it is more important to build software that meets the customers' needs today than worry about features that might be needed in the future. 38

敏捷开发的核心思想是快速交付能满足当前客户需求的软件，而不是提前规划和开发将来可能用得上的功能。这样可以更快响应变化，减少浪费。因此，题目中"更重要的是满足客户现在的需求，而不是担心将来可能需要的功能"这一说法正确.

(T) 5. 在敏捷开发中，构建能够满足客户当前需求的软件比担心未来可能需要的功能更为重要.

( ) 5-6. What are the four framework activities found in the Extreme Programming (XP) process model? 39

A. analysis, design, coding, testing
B. planning, analysis, design, coding
C. planning, analysis, coding, testing
D. planning, design, coding, testing

极限编程(XP)流程模型的四个核心框架活动是规划(planning)、设计(design)、编码(coding)和测试(testing)。XP强调快速、灵活的开发，这四个环节循环进行，不断提升软件质量和响应需求变化。选项D正好对应这四大核心活动.

(D) 6. 极限编程(XP)过程模型中包含哪四个框架活动?
A. 分析、设计、编码、测试
B. 计划、分析、设计、编码
C. 计划、分析、编码、测试
D. **计划、设计、编码、测试**

( ) 5-7. All agile process models conform to a greater or lesser degree to the principles stated in the "Manifesto for Agile Software Development". 40

所有敏捷过程模型在不同程度上都遵循了"敏捷软件开发宣言"中的原则。所有敏捷开发方法(如Scrum、XP、Kanban等)都是基于敏捷宣言的核心价值观和原则设计的，虽然具体实践上不同，但都体现了如重视个体和互动、响应变化、持续交付等敏捷理念.

(T) 7. 所有敏捷过程模型在不同程度上都遵循《敏捷软件开发宣言》中提出的原则.

( ) 5-8. Which is not one of the key questions that is answered by each team member at each daily Scrum meeting? 41

A. What did you do since the last meeting?
B. What obstacles are you encountering?
C. What obstacles are you encountering?
D. What do you plan to accomplish be the next team meeting?

每日Scrum会议关注三个核心问题：你昨天做了什么？你今天准备做什么？你遇到了什么障碍？选项B和C都问"你遇到了什么障碍？"，内容重复。每日Scrum不会重复提问，因此C不是每日Scrum需要单独提问的问题.

(C) 8. 以下哪一项不是每位团队成员在每日Scrum会议上需要回答的关键问题?
A. 自上次会议以来你做了什么?
B. 你遇到了哪些障碍?
C. **你遇到了哪些障碍?**
D. 你计划在下次团队会议前完成什么?

( ) 5-9. The Dynamic Systems Development Method (DSDM) suggests a philosophy that is based on the Pareto principle (80% of the application can be delivered in 20% of the time required to build the complete application). 42

DSDM(动态系统开发方法)强调用帕累托原则指导开发, 即"80%的应用功能可以在完成整个应用所需时间的20%内交付". DSDM鼓励先快速实现最重要的核心功能, 后续逐步完善剩余部分, 这样能尽早获得可用系统并及时响应变化.

(T) 9. 动态系统开发方法(DSDM)提出了一种基于帕累托原则的理念(80%的应用可以在构建完整应用所需时间的20%内交付).

( ) 5-10. Agile Modeling (AM) provides guidance to practitioner during which of these software tasks? 43

A. Analysis
B. Design
C. Coding
D. Testing

Agile Modeling(敏捷建模)主要在软件开发的分析和设计阶段提供指导, 强调快速、灵活地建立和改进模型, 支持需求分析和系统设计. 编码和测试阶段并不是AM的主要指导范围.

(AB) 10. 敏捷建模(AM)为从业者在以下哪些软件任务中提供指导?
A. **分析**
B. **设计**
C. 编码
D. 测试

( ) 5-11. Agile Unified Process uses the classic UP phased activities (inception, elaboration, construction, transition) to help the team visualize the overall process flow. 44

敏捷统一过程(Agile Unified Process, AUP)采用了统一过程(Unified Process, UP)的四个经典阶段: 初始(inception)、细化(elaboration)、构建(construction)、移交(transition), 帮助团队明确开发流程的结构和进展. 通过这些阶段, 团队可以把握项目的不同开发阶段和目标.

(T) 11. 敏捷统一过程(Agile Unified Process)使用经典的统一过程(UP)阶段活动(初始、细化、构建、移交)来帮助团队可视化整体流程.

( ) 6-1. Human aspects of software engineering are not relevant in today's agile process models. 45

敏捷开发高度重视人的因素, 如团队沟通、协作、反馈和自组织团队. 敏捷方法强调人与人之间的交流比流程和工具更重要, 因此人的因素在敏捷开发中非常关键.

(F) 1. 软件工程中的人为因素在当今的敏捷过程模型中并不重要.

( ) 6-2. Which of the following is not an important trait of an effective software engineer? 46

A. Attentive to detail
B. Brutally honest
C. Follows process rule dogmatically
D. Resilient under pressure

一个有效的软件工程师应注重细节、诚实、能在压力下保持韧性, 但"死板地遵循流程规则"不是重要特质. 软件工程过程需要灵活应对变化, 机械执行流程会影响项目进展和创新能力, 所以灵活性和适应性更重要.

(C) 2. 以下哪一项不是高效软件工程师的重要特质?
A. 注重细节
B. 极度诚实
C. **死板地遵循流程规则**
D. 在压力下有韧性

( ) 6-3. Group communication and collaboration are as important as the technical skills of an individual team member to the success of a team. 47

团队成员之间的沟通与协作和个人的技术能力一样, 对团队的成功非常重要. 在软件工程项目中, 仅有个人能力还不够, 良好的沟通和协作能帮助成员理解需求、协调工作、及时解决问题, 从而提升项目整体效率和质量.

(T) 3. 团队成员之间的群体沟通与协作与个人的技术能力同等重要, 对于团队的成功至关重要.

( ) 6-4. Teams with diversity in the individual team member skill sets tend to be more effective than teams without this diversity. 48

团队成员技能多样化可以集思广益, 成员之间互补, 有助于解决复杂问题, 提升创新能力和整体效率. 技能单一的团队容易有知识盲区, 难以应对多样化需求. 因此, 软件工程中通常鼓励组建技能互补的团队.

(T) 4. 拥有成员技能多样性的团队往往比没有这种多样性的团队更有效.

( ) 6-5. Which of the following can contribute to team toxicity? 49

A. Frenzied work atmosphere
B. Inadequate budget
C. Poorly coordinated software process
D. Unclear definition of team roles

A"紧张忙乱的工作氛围"会增加压力, 容易引发成员间摩擦; B"预算不足"让团队无法获得必要资源, 增加矛盾和不满; D"团队角色定义不清"会导致职责混乱、争执频发, 这三项都会直接影响团队合作氛围. C"软件过程协调不佳"主要影响项目进度和质量, 是流程问题, 不一定直接导致团队成员间的毒性氛围.

(ABD) 5. 下列哪些因素可能导致团队毒性?
A. **紧张忙乱的工作氛围**
B. **预算不足**
C. 软件过程协调不良
D. **团队角色定义不清**

( ) 6-6. Software engineering team structure is independent of problem complexity and size of the expected software products. 50

软件工程团队的结构会受到问题复杂度和预期软件产品规模的影响. 复杂问题和大型软件产品通常需要更专业化、更分层次或更灵活的团队结构, 而简单或小型项目可能采用更扁平、简单的团队结构, 因此团队结构不是独立于这些因素的.

(F) 6. 软件工程团队结构与问题的复杂性和预期软件产品的规模无关.

( ) 6-7. Agile teams are allowed to self-organize and make their own technical decisions. 51

敏捷团队被允许自我组织和自主做技术决策, 这是敏捷方法的核心原则之一. 团队成员根据项目需求和实际情况协作决定实现目标的方式, 而不是由外部管理者指定具体做法. 这样有助于提升团队积极性、责任感和创新能力.

(T) 7. 敏捷团队被允许自我组织并做出自己的技术决策.

( ) 6-8. In XP a metaphor is used as a device to facilitate communications among customers, team members, and managers? 52

在极限编程(XP)中, 隐喻(metaphor)被用作促进客户、团队成员和管理者之间沟通的工具. 在XP方法中, 团队选择一个简单、易理解的系统隐喻, 帮助各方对系统工作方式形成共同理解, 减少沟通障碍, 提高协作效率.

(T) 8. 在极限编程(XP)中, 隐喻被用作促进客户、团队成员和管理者之间沟通的工具吗?

( ) 6-9. Using an established social media platform negates the need to be concerned about privacy or security. 53

即使使用知名社交媒体平台, 仍然可能遇到隐私泄露和安全漏洞. 用户发布的信息可能被滥用、遭黑客攻击或被平台不当利用, 所以不管平台多有名, 隐私和安全问题都必须重视.

(F) 9. 使用已有的社交媒体平台就无需担心隐私或安全问题.

( ) 6-10. Use of cloud services can speed up information sharing among software team members? 54

云服务(如Google Drive、GitHub、Slack等)让团队成员能随时随地访问、编辑和同步文档、代码或其他项目资料, 提高信息传递和协作效率. 因此, 使用云服务能加快团队成员之间的信息共享.

(T) 10. 使用云服务可以加快软件团队成员之间的信息共享吗?

( ) 6-11. In collaborative development environments, metrics are used to reward and punish team members. 55

指标(metrics)在协作开发环境中, 主要用于评估项目进展、提升团队效率和发现改进点, 而不是用来奖励或惩罚个人. 将指标用于奖惩会导致行为扭曲, 影响团队合作氛围. 正确做法是用指标帮助团队优化流程和质量, 而不是作为个人奖惩依据.

(F) 11. 在协作开发环境中, 度量指标被用来奖励和惩罚团队成员.

( ) 6-12. Which of these factors complicate decision-making by global software teams 56

A. Complexity of problem
B. Different views of the problem
C. Law of unintended consequences
D. Risk associated with decision

全球软件团队成员来自不同国家和文化, 协作时会遇到以下问题: A. 问题本身的复杂性增加达成一致的难度; B. 背景不同导致对同一问题有不同看法, 决策更难; C. 决策可能带来预料之外的后果(意外后果定律), 增加复杂性; D. 每个决策都伴随风险, 需要评估和管理. 这四个因素都会让全球团队的决策过程更复杂.

(ABCD) 12. 以下哪些因素会使全球软件团队的决策变得复杂?
A. **问题的复杂性**
B. **对问题的不同看法**
C. **意外后果法则**
D. **与决策相关的风险**

( ) 7-1. Software engineering principles have about a three year half-life. 57

题目认为软件工程的原则每三年就会"折半"或失效一半. 实际上, 软件工程的基本原则(如模块化、抽象、可维护性等)相对稳定和持久, 不会在三年内过时或无效. 软件工程原则的有效期远远超过三年.

(F) 1. 软件工程原理的半衰期大约为三年.

( ) 7-2. Which of the following is not one of core principles of software engineering practice? 58

A. All design should be as simple as possible, but no simpler.
B. A software system exists only to provide value to its users.
C. Pareto principle (20% of any product requires 80% of the effort).
D. Remember that you produce others will consume

A、B、D都是软件工程实践的重要原则: A强调设计要尽量简单, B强调软件存在的价值在于为用户提供价值, D强调开发人员要考虑他人会使用自己的产出. C项"帕累托原则(20%的产品需要80%的努力)"主要属于管理或经济领域的经验法则, 不是软件工程实践的核心原则.

(C) 2. 以下哪一项不是软件工程实践的核心原则?
A. 所有设计都应尽可能简单, 但不能过于简单.
B. 软件系统的存在只是为了为其用户提供价值.
C. **帕累托原则(任何产品的20%需要80%的努力).**
D. 记住你所产出的将被他人使用.

( ) 7-3. Every communication activity should have a facilitator to make sure that the customer is not allowed to dominate the proceedings. 59

题目说每次沟通活动都应该有主持人，目的是防止客户主导会议，这种说法不对．主持人可以帮助会议有序进行，但不是每次都必须有主持人．更重要的是促进开放和平等的交流．客户主导并不总是负面的，有时客户的需求和意见很重要，应该被充分表达和重视．

(F) 3. 每一次交流活动都应该有一位协调者，以确保客户不会主导整个过程．

( ) 7-4. The agile view of iterative customer communication and collaboration is applicable to all software engineering practice. 60

敏捷方法强调持续与客户沟通和紧密协作，帮助团队及时理解客户需求的变化，提高软件质量和客户满意度．这种思想不仅适用于敏捷开发流程，也对需求分析、设计、测试等软件工程实践有积极影响，有助于减少误解和返工，因此适用于所有软件工程实践．

(T) 4. 敏捷方法中关于迭代式客户沟通与协作的观点适用于所有软件工程实践．

( ) 7-5. One reason to involve everyone on the software team in the planning activity is to 61

A. adjust the granularity of the plan
B. control feature creep
C. get all team members to “sign up” to the plan
D. understand the problem scope

让所有团队成员参与计划活动的原因，是为了使大家对计划达成共识，承诺完成各自任务(即“sign up”)．这样能增强成员的责任感和归属感，提高计划执行力．其他选项虽然重要，但不是让每个人参与计划活动的主要目的．

(C) 5. 让软件团队中的每个人都参与计划活动的一个原因是
A. 调整计划的细粒度
B. 控制功能蔓延
C. **让所有团队成员都“认同”该计划**
D. 理解问题范围

( ) 7-6. Project plans should not be changed once they are adopted by a team. 62

这句话意思是“项目计划一旦被团队采纳后就不应该更改．”实际上，项目计划在执行过程中经常需要根据实际情况调整，比如需求变更、进度延误、资源变化等．灵活调整计划有助于项目适应变化，保证按时高质量完成．因此，项目计划不是一成不变的．

(F) 6. 项目计划一旦被团队采纳后就不应更改．

( ) 7-7. Requirements models depict software in which three domains? 63

A. architecture, interface, component
B. cost, risk, schedule
C. information, function, behavior
D. None of the above

需求建模(requirements modeling)描述软件系统的三个主要方面：信息域(information domain，说明系统处理和存储的数据)、功能域(function domain，说明系统需实现的功能)、行为域(behavior domain，说明系统在不同条件下的响应和表现)．选项C正对应这三大领域．其他选项提到的是架构、接口、成本等内容，不属于需求模型的核心领域．

(C) 7. 需求模型描述软件的哪三个领域?
A. 架构、接口、组件
B. 成本、风险、进度
C. **信息、功能、行为**
D. 以上都不是

( ) 7-8. The design model should be traceable to the requirements model? 64

设计模型应能追溯到需求模型，以确保设计覆盖所有需求．缺乏追溯性可能导致需求遗漏或实现错误．追溯性也便于后期维护和变更时查找设计决策依据．

(T) 8. 设计模型应该可以追溯到需求模型吗?

( ) 7-9. Teams using agile software practices do not generally create models. 65

敏捷开发不是不建模，而是强调“适度建模”(Just Enough Modeling)．敏捷团队会根据需要创建简单、有效的模型，如用例图、流程图等，帮助团队达成共识和指导开发．与传统方法相比，敏捷方法不会做大量、详细的文档和模型，更注重快速交付和灵活应变．所以敏捷团队会建模，只是更注重效率和实用性．

(F) 9. 使用敏捷软件实践的团队通常不会创建模型.

( ) 7-10. Which of the following is not one of the principles of good coding? 66

A. Create unit tests before you begin coding
B. Create unit tests before you begin coding
C. Refractor the code after you complete the first coding pass
D. Write self-documenting code, not program documentation

A和B都指“在编码前创建单元测试”，这是测试驱动开发的推荐做法．D“编写自解释代码而不是依赖文档”也是现代开发中的提倡方向．C“在第一次编码完成后重构代码”有误，正确做法应是持续重构，而不是只在第一次编码后进行，因此C不属于良好编码原则．

(C) 10. 以下哪一项不是良好编码原则之一?
A. 在开始编码之前编写单元测试
B. 在开始编码之前编写单元测试
C. **在完成第一次编码后重构代码**
D. 编写自说明代码，而不是编写程序文档

( ) 7-11. A successful test I ones that discovers at least one as-yet undiscovered error. 67

软件测试的主要目标是发现错误，能找出新的缺陷说明测试有效．如果测试没有发现任何问题，可能是覆盖面或测试设计不够好．发现新的错误就是测试成功的标志．

(T) 11. 一次成功的测试是能够发现至少一个尚未被发现的错误的测试.

( ) 7-12. Which of the following are valid reasons for collecting customer feedback concerning delivered software? 68

A. Allows developers to make changes to the delivered increment
B. Delivery schedule can be revised to reflect changes
C. Developers can identify changes to incorporate into next increment
D. All of the above

收集客户反馈可以让开发人员了解用户对已交付软件的实际使用情况，从而：A项，帮助开发人员对已交付软件进行必要的改进和修正；B项，如果客户反馈需求有变化，交付计划可能需要调整以适应变化；C项，开发人员能根据反馈，在下一个增量开发时加入用户期望的新功能或修正问题．因此，以上三个选项都是收集客户反馈的有效理由．

(D) 12. 以下哪些是收集客户对已交付软件反馈的有效理由?
A. 允许开发人员对已交付的增量进行更改
B. 交付进度可以根据变更进行调整
C. 开发人员可以确定需要在下一个增量中纳入的更改
D. **以上都是**

( ) 7-13. Larger programming teams are always more productive than smaller teams. 69

团队规模变大并不一定提升生产力．人数增加会带来更多沟通和协调成本，可能反而降低效率．小团队通常更灵活，沟通顺畅，效率可能更高．所以大团队不一定比小团队更有生产力．

(F) 13. 更大的编程团队总是比小的团队更高效.

( ) 8-1. Requirements engineering is a generic process that does not vary from one software project to another. 70

需求工程(requirements engineering)有一些通用的基本步骤，比如需求获取、分析、规范和验证，但它不是完全统一的流程．不同项目因规模、复杂度、领域和用户需求不同，具体需求工程过程会根据实际情况调整和变化．因此，需求工程不是不变的通用过程．

(T) 1. 需求工程是一个通用过程，不会因不同的软件项目而变化.

( ) 8-2. During project inception the intent of the of the tasks are to determine 71

A. basic problem understanding
B. nature of the solution needed
C. people who want a solution
D. none of the above

项目初始阶段(inception)的任务主要包括：理解基本问题(A)、明确需要什么样的解决方案(B)、确定有哪些相关人员或利益相关者需要这个解决方案(C)．这三个方面有助于团队全面把握项目的基本情况和目标．

(ABC) 2. 在项目启动阶段，任务的目的是确定
A. **对基本问题的理解**
B. **所需解决方案的性质**
C. **希望获得解决方案的人**
D. 以上都不是

( ) 8-3. Three things that make requirements elicitation difficult are problems of 72

A. budgeting
B. scope
C. understanding
D. volatility

B(scope)指需求范围难以界定，容易出现范围蔓延；C(understanding)指需求人员与用户之间沟通不充分，导致对需求理解不一致；D(volatility)指需求经常变化，增加了获取难度．A(budgeting)是项目资金问题，不是需求获取的主要难点．

(BCD) 3. 使需求获取变得困难的三个因素是以下哪些问题
A. 预算
B. **范围**
C. **理解**
D. **易变性**

( ) 8-4. A stakeholder is anyone who will purchase the completed software system under development. 73

利益相关者(stakeholder)不仅包括购买软件系统的人，还包括所有对软件开发过程或结果有影响或受影响的人，如项目经理、开发人员、测试人员、最终用户、维护人员、投资方等．因此，利益相关者的范围远比“购买者”要广．

(F) 4. 利益相关者是指任何将购买正在开发的软件系统的人.

( ) 8-5. It is relatively common for different customers to propose conflicting requirements, each arguing that his or her version is the right one. 74

在需求获取阶段，不同客户因各自利益、立场或理解不同，经常会提出相互冲突的需求，每个人都坚持自己的需求才是正确的．这种情况在需求分析过程中很常见．

(T) 5. 不同客户提出相互冲突的需求是比较常见的，每个客户都认为自己的版本才是正确的.

( ) 8-6. Which of the following is not one of the context-free questions that would be used during project inception? 75

A. What will be the economic benefit from a good solution?
B. Who is behind the request for work?

C. Who will pay for the work?
D. Who will use the solution?

本题考查项目启动阶段常见的“上下文无关”问题，即在项目早期，不涉及具体实现细节时需问的基本问题。A、B、D选项关注项目背景、用户和需求，是典型的上下文无关问题；C选项“谁来为这项工作买单”涉及资金来源和具体商业安排，属于项目管理和执行细节，不属于上下文无关问题。
(C) 6. 下列哪一项不是在项目启动阶段会使用的无上下文问题?
A. 一个好的解决方案将带来什么经济效益?
B. 谁在提出这项工作的请求?
C. **谁将为这项工作买单?**
D. 谁将使用该解决方案?

( ) 8-7. Non-functional requirements can be safely ignored in modern software development projects. 76

非功能性需求(Non-functional requirements)包括性能、安全性、可用性、可靠性等，直接影响系统质量和用户体验。忽略这些需求可能导致软件无法满足用户或业务实际需求，甚至引发安全和法律风险。因此，软件开发过程中必须重视非功能性需求的分析和实现。
(F) 7. 在现代软件开发项目中，非功能性需求可以被安全地忽略。

( ) 8-8. In collaborative requirements gathering the facilitator 77

A. arranges the meeting place
B. can not be a customer
C. controls the meeting
D. must be an outsider

facilitator(引导者)在协作式需求收集中的主要职责是控制会议，保证讨论有序高效，引导各方发言，推动会议达成目标。安排会议地点、不能是客户或必须是外部人员，并不是facilitator的核心职责或必要条件。
(C) 8. 在协作需求收集过程中，协调者
A. 安排会议地点
B. 不能是客户
C. **控制会议**
D. 必须是外部人员

( ) 8-9. Which of the following is not one of the requirement classifications used in Quality Function Deployment (QFD)? 78

A. exciting
B. expected
C. mandatory
D. normal

QFD(质量功能展开)中的需求分类有 expected(期望的)、normal(正常的/显性的)、exciting(令人兴奋的/魅力型)，这些分类来源于Kano模型，分别对应用户的基本需求、明确需求和能带来惊喜的需求。mandatory(强制性的)不是QFD正式的需求分类。
(C) 9. 以下哪一项不是质量功能展开(QFD)中使用的需求分类?
A. 令人兴奋的
B. 预期的
C. **强制性的**
D. 正常的

( ) 8-10. The work products produced during requirement elicitation will vary depending on the 79

A. size of the budget.
B. size of the product being built.
C. software process being used.
D. stakeholders needs.

需求获取阶段的工作产品(如需求规格说明、用例、用户故事等)，会根据产品的规模(B)和干系人的需求(D)而变化。产品越大、越复杂，需求文档就可能越详细，工作产品也会更多样化。不同干系人的需求也直接影响需求获取结果的内容和形式。预算大小(A)虽然可能间接影响需求获取的深度或资源分配，但不是直接决定工作产品类型和内容的主要因素。软件过程(C)主要影响开发流程和活动，不是工作产品多样化的首要因素。正确答案应该是B和D。
(AB) 10. 在需求获取过程中产生的工作产品会根据以下哪些因素而变化
A. **预算的规模.**
B. **所开发产品的规模.**
C. 所采用的软件过程.
D. 相关方的需求.

( ) 8-11. User stories are complete descriptions the user needs and include the non-functional requirements for a software increment. 80

User stories(用户故事)通常只是对用户需求的简短描述，关注用户想做什么和为什么做，但不包含完整需求描述，尤其很少包括非功能性需求(如性能、安全性等)。非功能性需求一般需通过单独条目或文档补充说明。所以，用户故事不是用户需求的完整描述，也通常不包括非功能性需求。
(T) 11. 用户故事是对用户需求的完整描述，并包含软件增量的非功能性需求.

( ) 8-12. Developers and customers create use-cases to help the software team understand how different classes of end-users will use functions. 81

开发者和客户共同创建用例，用于帮助软件团队理解不同类型的最终用户如何使用软件功能。用例本质上是对系统功能的场景描述，通常由开发者和客户合作编写，以确保软件满足用户实际需求。通过用例，团队能够更好地了解各种用户与系统的交互方式，有助于需求分析和后续设计开发。
(T) 12. 开发人员和客户创建用例，以帮助软件团队了解不同类别的最终用户将如何使用功能.

( ) 8-13. Use-case actors are always people, never system devices. 82

用例图中的参与者(actor)可以是人，也可以是其他系统或设备。任何与系统有交互的外部实体，无论是人、硬件设备还是其他软件系统，都可以作为参与者。所以参与者不一定是人，也可以是系统设备。
(F) 13. 用例的参与者总是人，而不是系统设备.

( ) 8-14. The result of the requirements engineering task is an analysis model that defines which of the following problem domain(s)? 83

A. information
B. functional
C. behavioral
D. all of the above

需求工程的最终成果是分析模型，该模型全面描述问题域，包括信息域(系统处理和存储的信息)、功能域(系统需要实现的功能)、行为域(系统对事件的响应和状态变化)。因此，分析模型覆盖信息、功能和行为这三个方面。
(D) 14. 需求工程任务的结果是一个分析模型，该模型定义了以下哪些问题域?
A. 信息域
B. 功能域
C. 行为域
D. **以上所有**

( ) 8-15. Analysis patterns facilitate the transformation of the analysis model into a design model by suggesting reliable solutions to common problems. 84

分析模式(analysis patterns)通过为常见问题提供可靠的解决方案，帮助将分析模型转化为设计模型。分析模式总结了分析阶段常见问题及其解决思路，开发人员可以直接应用这些成熟模式，更高效、规范地将需求分析结果过渡到系统设计阶段。
(T) 15. 分析模式通过为常见问题提供可靠的解决方案，促进了分析模型向设计模型的转化.

( ) 8-16. In agile process models requirements engineering and design activities are interleaved. 85

在敏捷过程模型中，需求工程和设计活动是交错进行的，不是先全部完成需求再做设计。敏捷强调迭代开发，每一步都根据最新需求及时调整设计，二者同步推进、相互影响。这有助于快速响应变化，提高开发效率。
(T) 16. 在敏捷过程模型中，需求工程和设计活动是交错进行的.

( ) 8-17. In win-win negotiation, the customer's needs are met even though the developer's need may not be. 86

“双赢谈判”(win-win negotiation)强调客户和开发者双方的需求都要被合理满足，达到双方满意的结果。如果只有客户的需求被满足，开发者的需求被忽视，这不属于双赢，而是单方面让步。因此，题目的说法错误。
(F) 17. 在双赢谈判中，即使开发者的需求可能没有被满足，客户的需求也会被满足.

( ) 8-18. In requirements validation the requirements model is reviewed to ensure its technical feasibility. 87

requirements validation(需求验证)主要检查需求是否正确、完整、一致、可理解，以及是否满足用户需求，不涉及技术可行性的评估。技术可行性通常在需求分析后、系统设计前的可行性研究阶段进行。因此，题目中关于需求验证确保技术可行性的说法是错误的。
(F) 18. 在需求验证中，会对需求模型进行评审，以确保其技术可行性.

( ) 8-19. The most common reason for software project failure is lack of functionality. 88

软件项目失败最常见的原因不是功能不足，而是项目管理不善，如需求不明确、进度控制不当、沟通不畅等。虽然功能重要，但失败通常因为未能按时、按预算交付，或产品质量不符合用户期望。项目管理和需求管理是决定项目成败的关键因素。
(F) 19. 软件项目失败最常见的原因是功能缺失.

( ) 9-1. Which of these is not an element of a requirements model? 89

A. Behavioral elements
B. Class-based elements
C. Data elements
D. Scenario-based elements

需求模型(requirements model)用于描述系统应实现的功能和行为。其常见要素有：行为元素(Behavioral elements)，描述系统在不同条件下的行为；类元素(Class-based elements)，描述系统中的类及其关系；场景元素(Scenario-based elements)，通过用例等描述用户与系统的交互场景。数据元素(Data elements)虽然属于系统分析内容，但不是需求模型的核心组成部分，更侧重于数据库设计等。
(C) 1. 以下哪一项不是需求模型的组成元素?
A. 行为元素
B. 基于类的元素
C. **数据元素**
D. 基于场景的元素

( ) 9-2. Which of the following is not an objective for building a requirements model? 90

A. define set of software requirements that can be validated
B. describe customer requirements
C. develop an abbreviated solution for the problem
D. establish basis for software design

需求建模的主要目标是明确、描述和验证客户需求，并为后续软件设计提供基础。选项C“为问题开发一个简化的解决方案”并不是需求建模的目标。需求建模关注“要做什么”，而不是“怎么做”或提供具体解决方案，所以C不是需求建模的目标。
(C) 2. 以下哪一项不是构建需求模型的目标?
A. 定义一组可以验证的软件需求
B. 描述客户需求
C. **为问题开发一个简化的解决方案**
D. 为软件设计建立基础

( ) 9-3. Object-oriented domain analysis is concerned with the identification and specification of reusable capabilities within an application domain. 91

面向对象领域分析的主要目标是在特定应用领域中识别和规范可复用的功能(能力或组件). 通过分析领域中的共性和变异点, 提取可以被多个系统或项目复用的类、对象及其关系, 从而提高软件开发的效率和质量. 这种分析有助于建立可重用的领域模型, 为后续的软件设计和实现打下基础.
(T) 3. 面向对象的领域分析关注于在应用领域内可重用能力的识别和规范.

( ) 9-4. In structured analysis models focus on the structure of the classes defined for a system along with their interactions. 92

结构化分析(structured analysis)主要关注系统的功能、数据流和过程, 不关注类的结构和交互. 类的结构和交互是面向对象分析(object-oriented analysis)关注的内容, 因此结构化分析不强调类的结构.
(F) 4. 在结构化分析中, 模型侧重于为系统定义的类的结构及其交互.

( ) 9-5. Creation and refinement of use cases if an important part of scenario-based modeling. 93

用例的创建和完善是基于场景建模的重要部分, 因为用例描述了系统与用户之间的交互场景, 有助于理解需求和系统行为. 通过不断细化用例, 可以更准确地描述系统的功能和用户需求, 为后续的软件设计和实现打下基础.
(T) 5. 用例的创建和完善是基于场景建模的重要部分.

( ) 9-6. It is important to consider alternative actor interactions when creating a preliminary use case. 94

在创建初步用例时, 主要关注核心的、正常的交互流程, 即基本成功场景. 替代性参与者交互(如异常流程或替代路径)通常在用例进一步细化和扩展时再考虑, 而不是在最初阶段. 因此, 初步用例阶段不需要详细考虑所有替代性交互.
(F) 6. 在创建初步用例时, 考虑备选参与者交互是很重要的.

( ) 9-7. Brainstorming is one technique that may be used to derive a complete set of use case exceptions. 95

头脑风暴是一种集体讨论技术, 能让团队成员自由提出各种可能的异常情况, 从而更全面地识别和补充用例中的异常流程. 因此, 头脑风暴适合用来推导用例异常.
(T) 7. 头脑风暴是一种可用于推导完整用例异常集的技术.

( ) 9-8. In many cases there is no need to create a graphical representation of a usage scenario. 96

在软件工程中, 用例图有助于理解系统需求, 但不是每个用例场景都必须画成图. 文字描述有时已经足够, 尤其在场景简单或团队沟通顺畅时. 图形化只在需要更直观表达时使用, 并非必须.
(T) 8. 在许多情况下, 没有必要为使用场景创建图形化表示.

( ) 9-9. UML activity diagrams are useful in representing which analysis model elements? 97

A. Behavioral elements
B. Class-based elements
C. Flow-based elements
D. Scenario-based elements

UML活动图主要用于表示系统中的场景流程, 对某一用例或业务流程的具体步骤和分支进行建模. 它们适合描述以场景为基础的分析模型元素, 如用例的执行过程、用户和系统的交互步骤等.
(D) 9. UML活动图在表示哪些分析模型元素时是有用的?
A. 行为元素
B. 基于类的元素
C. 基于流程的元素
D. **基于场景的元素**

( ) 9-10. UML swimnlane diagrams allow you to represent the flow of activities by showing the actors having responsibility for creating each data element. 98

UML泳道图(Swimlane Diagram)主要用于表示活动流程, 并通过泳道区分不同参与者(如角色或部门), 显示每项活动由谁负责. 泳道图关注活动的执行者和流程, 而不是专门展示谁负责创建每个数据元素, 因此它不侧重于数据元素的创建责任, 而是反映活动流程和责任分工.
(F) 10. UML泳道图允许你通过显示对创建每个数据元素负责的参与者, 来表示活动的流程.

( ) 10-1. Which of the following should be considered as candidate objects in a problem space? 99

A. events
B. people
C. structures
D. all of the above

面向对象分析中, 候选对象是在问题空间中可能建模为"对象"的实体, 包含事件(如订单创建)、人(如用户、管理员)和结构(如数据库表、文档等). 分析问题时, 这三类都应被视为可能的对象.
(D) 1. 以下哪些应被视为问题空间中的候选对象?
A. 事件
B. 人员
C. 结构
D. **以上全部**

( ) 10-2. In the grammatical parse of a processing narrative the nouns become object candidates in the analysis model. 100

面向对象分析中, 处理需求描述时通过语法解析, 名词通常代表系统中的事物或概念, 因此作为分析模型中的对象候选项. 我们会把文本中的名词作为可能的对象来考虑, 这是面向对象分析常用的方法.
(T) 2. 在处理叙述的语法解析中, 名词成为分析模型中的对象候选项.

( ) 10-3. Attributes are chosen for an object by examining the problem statement and identifying the entities that appear to be related. 101

属性(attributes)是描述对象(object)特征的数据, 不是只通过识别问题描述中相关实体来选取. 实体通常用于确定需要哪些对象, 而对象的属性要通过分析这些对象的具体特征及它们在系统中的作用来决定, 仅凭实体间的关联不能确定合适的属性. 属性的选择需要深入理解对象本身, 而不是只考虑实体之间的关系.
(F) 3. 通过检查问题陈述并识别出似乎相关的实体来为对象选择属性.

( ) 10-4. Which of the following is not one of the broad categories used to classify operations? 102

A. computation
B. data manipulation
C. event monitors
D. transformers

常见的操作分类有: computation(计算)、data manipulation(数据操作)和event monitors(事件监视器), 分别对应不同功能. "transformers"不是常用的操作分类术语, 因此不属于操作的广泛分类.
(D) 4. 下列哪一项不是用于分类操作的广泛类别之一?
A. 计算
B. 数据操作
C. 事件监控器
D. **转换器**

( ) 10-5. Collaborators in CRC modeling are those classes needed to fulfill a responsibility on another card. 103

CRC(Class-Responsibility-Collaborator)建模中的Collaborator(协作者)是指帮助当前类完成其责任的其他类. 当一个类需要另一个类来实现某项责任时, 这个被需要的类就是Collaborator. 因此, Collaborators是在CRC建模中为实现责任而需要的其他类.
(T) 5. 在CRC建模中, 协作者是指为实现另一张卡片上的某个职责而需要的类.

( ) 10-6. Which of the following items does not appear on a CRC card? 104

A. class collaborators
B. class name
C. class reliability
D. class responsibilities

CRC卡(Class-Responsibility-Collaborator卡)用于设计和理解面向对象系统, 卡片上包含类名(class name)、类的职责(class responsibilities)、以及与该类协作的其他类(class collaborators). 类的可靠性(class reliability)不是CRC卡的内容.
(C) 6. 下列哪一项不会出现在CRC卡片上?
A. 类的协作对象
B. 类名
C. **类的可靠性**
D. 类的职责

( ) 10-7. Class responsibilities are defined by 105

A. its attributes only
B. its collaborators
C. its operations only
D. both its attributes and operations

类的职责包括类需要完成的功能和任务. 职责不仅由类的数据(属性)决定, 还由它能执行的行为(操作/方法)决定. 只有属性(描述状态)和操作(描述行为)结合, 才能完整定义一个类的职责.
(D) 7. 类的职责由什么定义?
A. 仅由其属性
B. 由其协作者
C. 仅由其操作
D. **由其属性和操作**

( ) 10-8. A stereotype is the basis for class reuse in UML modeling. 106

在UML建模中, stereotype(构造型)是一种扩展机制, 用于为模型元素添加新的属性或语义, 并不是类复用的基础. 类复用通常通过继承、接口实现等机制实现, 而不是通过stereotype.
(F) 8. 构造型是UML建模中类重用的基础.

( ) 10-9. An analysis package involves the categorization of analysis model elements into useful groupings. 107

分析包(analysis package)是把分析模型中的元素按照标准分类, 分成有意义的组, 有助于管理和理解系统结构, 也方便后续设计和开发.
(T) 9. 分析包涉及将分析模型元素分类为有用的分组.

( ) 11-1. The behavior modeling is only used in the analysis of real-time systems. 108

行为建模不仅用于实时系统分析. 它是一种描述和理解系统动态行为的方法, 广泛应用于各种类型的软件系统, 包括非实时系统. 通过行为建模, 可以描述系统的状态变化、事件响应和交互过程, 所以不只局限于实时系统.
(F) 1. 行为建模仅用于实时系统的分析.

( ) 11-2. For purposes of behavior modeling an event occurs whenever 109

A. a state and process exchange information.
B. the system an actor exchange information.
C. two actors exchange information.
D. two objects exchange information.

行为建模中的“事件”(event)指系统与外部参与者(actor)之间的信息交换，如用户点击按钮、系统发送提示等。A选项是状态和过程，C选项是两个参与者，D选项是两个对象，这些都不符合行为建模中事件的定义。B选项强调了系统与参与者的信息交互，这是行为建模中事件发生的典型场景。

(B) 2. 在行为建模中，事件发生的情况是
A. 一个状态和一个过程交换信息.
B. **系统和参与者交换信息.**
C. 两个参与者交换信息.
D. 两个对象交换信息.

( ) 11-3. For purposes of behavior modeling a state is any 110

A. consumer or producer of data.
B. data object hierarchy.
C. observable mode of behavior.
D. well defined process.

在行为建模(behavior modeling)中，“状态”(state)指的是系统或对象在某一时刻表现出的可观察的行为模式，如“等待输入”“处理数据”“输出结果”等。A描述的是数据的生产和消费，B关注数据对象的层次结构，D指的是过程，都不是状态的本质含义。只有“可观察的行为模式”准确表达了状态在行为建模中的含义。

(C) 3. 在行为建模中，状态是指任何
A. 数据的消费者或生产者.
B. 数据对象的层次结构.
C. **可观察到的行为模式.**
D. 明确定义的过程.

( ) 11-4. The state transition diagram 111

A. depicts relationships between data objects
B. depicts functions that transform the data flow
C. indicates how data are transformed by the system
D. indicates system reactions to external events

状态转换图(state transition diagram)用来描述系统在接收到外部事件时，从一个状态转移到另一个状态的过程，即系统对外部事件的反应。其他选项分别涉及数据对象关系、数据流功能转换和数据变换，这些不是状态转换图的核心内容。

(D) 4. 状态转换图
A. 描述数据对象之间的关系
B. 描述转换数据流的功能
C. 表示数据如何被系统转换
D. **表示系统对外部事件的反应**

( ) 11-5. The UML sequence diagram shows the order in which system events are processed. 112

UML顺序图(sequence diagram)主要展示对象之间的消息传递顺序，描述对象或类之间的方法调用和交互过程，不是用来表示系统事件(如用户输入、外部系统信号等)被处理的顺序。顺序图关注的是对象之间的交互流程，而不是系统整体事件的处理顺序。

(F) 5. UML序列图显示了系统事件被处理的顺序.

( ) 11-6. Analysis patterns are discovered, they are not explicitly created. 113

分析模式(analysis patterns)是在分析过程中被发现的，而不是像设计模式那样有意识地创造出来。分析模式是对某一领域建模时常见、可复用的业务结构或关系，来源于对实际业务的反复分析和经验总结，而不是凭空设计。分析模式是被发现的。

(T) 6. 分析模式是被发现的，而不是被明确创建的.

( ) 11-7. It is not possible to justify the time required for mobile app requirements analysis. 114

移动应用需求分析很重要，因为它能明确用户需求和系统要解决的问题，避免后期返工和资源浪费。因此，投入时间在需求分析上是合理的，有助于提升开发效率和软件质量。

(F) 7. 无法证明移动应用需求分析所需时间的合理性.

( ) 11-8. Which is not one of the analysis activities that is used to create a complete analysis model? 115

A. Configuration analysis
B. Content analysis
C. Functional analysis
D. Market analysis

分析模型关注软件系统的需求、功能和结构。配置分析、内容分析、功能分析都与系统需求和功能实现相关，是分析模型的一部分。市场分析主要涉及市场需求、竞争对手、用户趋势等商业层面，和软件系统功能、结构分析无关，因此不属于分析模型的活动。

(D) 8. 以下哪一项不是用于创建完整分析模型的分析活动?
A. 配置分析
B. 内容分析
C. 功能分析
D. **市场分析**

( ) 11-9. Content objects are extracted from use cases by examining the scenario description for direct or indirect content references. 116

内容对象(content objects)是通过分析用例的场景描述，寻找其中直接或间接提到的数据、信息或实体来提取的。这有助于后续的建模和系统设计。

(T) 9. 通过检查用例场景描述中的直接或间接内容引用，可以从用例中提取内容对象.

( ) 11-10. What are the elements of a WebApp interaction model? 117

A. activity diagrams, sequence diagrams, state diagrams, interface prototype
B. activity diagrams, collaboration diagrams, sequence diagrams, state diagrams
C. use-cases, sequence diagrams, state diagrams, interface prototype
D. use-cases, sequence diagrams, state diagrams, sequence diagrams

WebApp交互模型的核心要素包括：用例(use-cases)描述用户与系统的交互需求；顺序图(sequence diagrams)和状态图(state diagrams)表现系统内部流程和对象状态变化；界面原型(interface prototype)直观展示用户界面设计。这些要素共同刻画了Web应用中用户与系统的交互方式。选项C涵盖了这些关键方面.

(C) 10. Web应用交互模型的要素有哪些?
A. 活动图、顺序图、状态图、界面原型
B. 活动图、协作图、顺序图、状态图
C. **用例、顺序图、状态图、界面原型**
D. 用例、顺序图、状态图、顺序图

( ) 11-11. UML activity diagrams can be used to represent the user observable functionality delivered by the WebApp as well as the operations contained in each analysis class. 118

UML活动图(activity diagrams)可以用来描述Web应用中用户可见和可操作的功能流程，也能细化到分析类中的具体操作步骤。它们适用于表达系统的行为流程，包括整体功能和类内部逻辑，因此题目中的两种情况都能用活动图建模.

(T) 11. UML活动图可用于表示Web应用程序所提供的用户可见功能，以及每个分析类中包含的操作.

( ) 11-12. Configuration analysis focuses on the architecture of the user's web browsing environment. 119

配置分析(Configuration analysis)关注软件系统本身的配置，包括版本管理、配置项、配置状态等，不涉及用户网页浏览环境的架构。题目混淆了配置分析和用户环境.

(F) 12. 配置分析侧重于用户网页浏览环境的体系结构.

( ) 12-1. Which of the following are areas of concern in the design model? 120

A. architecture
B. data
C. interfaces
D. project scope

设计模型关注软件的结构(architecture)、数据的组织与管理(data)、各模块及与外部的接口(interfaces)，这些是软件设计需要明确和实现的内容。项目范围(project scope)属于项目管理，不是设计模型的部分.

(ABC) 1. 以下哪些是设计模型中关注的领域?
A. **架构**
B. **数据**
C. **接口**
D. 项目范围

( ) 12-2. The importance of software design can be summarized in a single word 121

A. accuracy
B. complexity
C. efficiency
D. quality

软件设计的核心意义是保证最终软件的质量，包括可维护性、可扩展性、可靠性等方面。准确性、效率和复杂性虽然也是软件设计关注的内容，但它们都属于影响软件质量的因素。软件设计的最终目标是提升和保证软件的整体质量.

(D) 2. 软件设计的重要性可以用一个词来概括
A. 准确性
B. 复杂性
C. 效率
D. **质量**

( ) 12-3. Which of these are characteristics of a good design? 122

A. exhibits strong coupling between its modules
B. implements all requirements in the analysis model
C. includes test cases for all components
D. provides a complete picture of the software

B选项“实现分析模型中的所有需求”正确，好的设计必须覆盖并实现分析阶段确定的所有需求，确保软件功能完整。D选项“提供软件的完整图景”也正确，好的设计应全面描述系统结构和行为，便于开发和维护人员理解系统。A选项“表现为模块间强耦合”错误，好的设计应模块间低耦合、高内聚，便于维护和扩展。C选项“为所有组件包括测试用例”不属于设计阶段的特征，测试用例是测试阶段的产物.

(BD) 3. 以下哪些是良好设计的特征?
A. 模块之间表现出强耦合
B. **实现了分析模型中的所有需求**
C. 为所有组件都包含测试用例
D. **提供了软件的完整图景**

( ) 12-4. Which of the following is not a characteristic common to all design methods? 123

A. configuration management
B. functional component representation
C. quality assessment guidelines
D. refinement heuristics

题目考查设计方法的共同特征。B(功能组件表示)、C(质量评估指导)、D(逐步细化的启发式方法)是各种设计方法普遍具备的特征。A(配置管理)指对软件变更进行控制和管理，并不是所有设计方法必需具备的特征，更偏向于过程和项目管理，因此A不是所有设计方法的共同特征.

(A) 4. 下列哪一项不是所有设计方法共有的特征?
A. **配置管理**
B. 功能组件表示

C. 质量评估指南
D. 细化启发式

( ) 12-5. What types of abstraction are used in software design? 124

A. control
B. data
C. environmental
D. procedural

软件设计中常用的抽象类型有控制抽象(control abstraction)、数据抽象(data abstraction)和过程抽象(procedural abstraction). 控制抽象是隐藏具体的控制流程细节, 如用流程结构或模块表达流程; 数据抽象只关注数据的逻辑特性, 隐藏具体实现, 比如类和数据结构; 过程抽象是把一组操作封装成过程或函数, 只看功能不关心实现细节. environmental abstraction(环境抽象)不是常用的抽象类型.

(ABD) 5. 软件设计中使用了哪些类型的抽象?
A. **控制抽象**
B. **数据抽象**
C. 环境抽象
D. **过程抽象**

( ) 12-6. Which of the following can be used to represent the architectural design of a piece of software? 125

A. Dynamic models
B. Functional models
C. Structural models
D. All of the above

软件体系结构设计可用多种模型描述: 动态模型表现系统运行时行为(如状态转移图), 功能模型描述系统要完成的功能(如数据流图), 结构模型描述系统各部分的组织结构和关系(如类图). A、B、C三种模型都可以用于表示软件的体系结构设计.

(D) 6. 下列哪一项可以用来表示一段软件的体系结构设计?
A. 动态模型
B. 功能模型
C. 结构模型
D. **以上全部**

( ) 12-7. Design patterns are not applicable to the design of object-oriented software? 126

设计模式(Design Patterns)是为了解决面向对象软件设计中常见问题而提出的通用解决方案. 它们有助于系统变得更加灵活、可维护和可复用, 是面向对象设计的重要方法, 非常适用于面向对象软件的设计.

(F) 7. ~~设计模式不适用于面向对象软件的设计?~~

( ) 12-8. Since modularity is an important design goal it is not possible to have too many modules in a proposed design. 127

虽然模块化是软件设计的重要目标, 但模块数量过多会导致系统复杂性增加、模块间通信成本变高、管理难度增加. 好的设计应在模块数量、系统可维护性和复杂性之间平衡, 模块不是越多越好.

(F) 8. 由于模块化是一个重要的设计目标, 因此在设计方案中不可能存在过多的模块.

( ) 12-9. Information hiding makes program maintenance easier by hiding data and procedure from unaffected parts of the program. 128

信息隐藏(information hiding)是指将模块内部的数据和实现细节对外部模块隐藏, 只暴露必要接口. 这样, 程序其他部分不需要关心模块内部实现, 修改模块内部时只要接口不变, 其他模块不会受到影响, 从而简化了程序维护. 题目中的表述"通过隐藏数据和过程, 使未受影响的程序部分不受影响"正是信息隐藏的作用.

(T) 9. 信息隐藏通过将数据和过程隐藏在程序未受影响的部分, 从而使程序维护变得更容易.

( ) 12-10. Cohesion is a qualitative indication of the degree to which a module 129

A. can be written more compactly.
B. focuses on just one thing.
C. is able to complete its function in a timely manner.
D. is connected to other modules and the outside world.

内聚性(Cohesion)是指一个模块内部各元素为完成单一功能而协作的程度. 高内聚的模块专注于一件事, 功能单一明确, 有利于理解、测试和维护. 选项B准确描述了高内聚模块的特点, 其余选项与内聚性无关.

(B) 10. 内聚是对一个模块内各部分之间紧密程度的定性指示.
A. 能否更紧凑地编写.
B. **是否只专注于一件事.**
C. 是否能够及时完成其功能.
D. 是否与其他模块和外部世界相连接.

( ) 12-11. Coupling is a qualitative indication of the degree to which a module 130

A. can be written more compactly.
B. focuses on just one thing.
C. is able to complete its function in a timely manner.
D. is connected to other modules and the outside world.

耦合度(Coupling)指一个模块与其他模块或外部世界之间连接和依赖的程度. 耦合度高表示模块之间联系紧密, 改动一个模块可能影响其他模块. 理想情况下, 模块之间的耦合度应尽量低, 便于独立开发、测试和维护. D选项"is connected to other modules and the outside world"准确描述了耦合度的含义.

(D) 11. 耦合是对一个模块与其他模块之间联系程度的定性指示.
A. 能否更简洁地编写.
B. 是否只关注一件事.
C. 是否能够及时完成其功能.
D. **与其他模块及外部世界的连接程度.**

( ) 12-12. When using structured design methodologies the process of stepwise refinement is unnecessary. 131

题目问在结构化设计方法中, 逐步求精的过程是否不必要. 实际上, 逐步求精(stepwise refinement)是结构化设计的核心思想之一. 它是将复杂系统分解为更简单、易于理解的小部分, 逐步细化每个部分的实现细节. 如果没有逐步求精, 结构化设计无法有效分解和简化问题, 也难以实现高质量的软件设计. ~~因此, 在结构化设计方法中, 逐步求精是必不可少的.~~

(F) 12. ~~在使用结构化设计方法时, 逐步细化的过程是没有必要的.~~

( ) 12-13. Software designs are refactored to allow the creation of software that is easier to integrate, easier to test, and easier to maintain. 132

软件设计的重构(refactoring)是在不改变软件外部行为的前提下改进其内部结构. 重构使代码更清晰、更模块化, 便于集成新功能、进行测试和后续维护, 从而提升软件的可集成性、可测试性和可维护性.

(T) 13. 软件设计经过重构, 以便创建更易于集成、更易于测试和更易于维护的软件.

( ) 12-14. Which of the following is not one of the five design class types 133

A. Business domain classes
B. Entity classes
C. Process classes
D. User interface classes

软件设计常见的五种设计类类型是: 用户界面类(User interface classes)、业务域类(Business domain classes)、进程类(Process classes)、持久类(Persistent classes)、控制类(Control classes). B选项"实体类(Entity classes)"虽然与业务域类类似, 但在标准的五类设计类中没有"实体类"这个说法, 因此不属于五种设计类类型.

(B) 14. 以下哪一项不是五种设计类类型之一?
A. 业务领域类
B. **实体类**
C. 过程类
D. 用户界面类

( ) 12-15. Which design model elements are used to depict a model of information represented from the user's view? 134

A. Architectural design elements
B. Component-level design elements
C. Data design elements
D. Interface design elements

设计模型中的数据设计元素(Data design elements)关注系统中信息的结构、存储方式和流动路径, 直接反映用户关心的数据内容和组织形式, 因此用于从用户视角对信息建模. 其他选项如架构、组件或接口设计主要关注系统结构和交互方式, 而不是信息本身的表示.

(C) 15. 哪些设计模型要素用于从用户视角描述信息模型?
A. 架构设计要素
B. 组件级设计要素
C. **数据设计要素**
D. 界面设计要素

( ) 12-16. Which design is equivalent to the floor plan of a house? 135

A. Architectural design
B. Component-level design
C. Data design
D. Interface design

本题考查软件设计的不同层次. 建筑设计(Architectural design)在软件工程中类似房子的平面图, 主要描述系统的总体结构和各部分间的关系, 就像房子的各个房间和通道的布局. 组件级设计、数据设计、接口设计分别对应实现细节、数据结构和模块间交互, 不等同于整体结构布局.

(A) 16. 哪种设计相当于房屋的平面图?
A. **架构设计**
B. 构件级设计
C. 数据设计
D. 接口设计

( ) 12-17. Which design model is equivalent to the detailed drawings of the access points and external utilities for a house? 136

A. Architectural design
B. Component-level design
C. Data design
D. Interface design

"access points and external utilities for a house"指的是房子与外部世界的连接, 如门、窗、水电接口等, 这些对应软件系统中与外部交互的部分. 软件工程中的Interface design(接口设计)专门描述系统与外部系统或用户的交互方式, 相当于"出入口".

(D) 17. 哪种设计模型相当于房屋接入点和外部设施的详细图纸?
A. 架构设计
B. 构件级设计
C. 数据设计
D. **接口设计**

( ) 12-18. Which design model is equivalent to a set of detailed drawings for each room in a house? 137

A. Architectural design
B. Component-level design
C. Data design
D. Interface design

B选项Component-level design(构件级设计)相当于为房子的每个房间画详细图纸，因为它关注系统中每个小部分(如模块或类)的具体实现细节. Architectural design对应房子的总体布局图，Data design关注数据结构，Interface design关注模块之间的接口. 房间详细图纸最类似于B.

(B) 18. 哪种设计模型相当于为房子中每个房间绘制的详细图纸?
A. 架构设计
B. **构件级设计**
C. 数据设计
D. 接口设计

( ) 12-19. The deployment design elements specify the build order for the software components. 138

部署设计元素描述软件如何在硬件或网络环境中分布和运行，比如服务器、节点、通信方式等. 构建顺序(build order)指的是软件组件在编译和集成时的先后顺序，这是构建管理或实现阶段的内容，不属于部署设计的范围. 因此，部署设计元素不会指定软件组件的构建顺序.

(F) 19. 部署设计元素指定了软件组件的构建顺序.

( ) 13-1. The best representation of system architecture is an operational software prototype. 139

系统架构通常用架构图、模型或文档来描述系统各部分及其关系，而不是用可运行的软件原型. 原型主要用于验证需求或设计思路，不适合正式表达系统架构. 系统架构需要清晰、抽象、全面地展示结构，而不是具体实现细节.

(F) 1. 系统架构的最佳表示方式是一个可运行的软件原型.

( ) 13-2. The architectural representations can be an enabler for communication among project stakeholders. 140

架构表示(architectural representations)是指描述软件系统整体结构和组件关系的图表、文档等方式. 这些表示帮助项目相关人员(如开发者、测试人员、客户和管理者)直观理解系统设计和各部分协作，有助于促进各方沟通.

(T) 2. 架构表示可以促进项目相关方之间的沟通.

( ) 13-3. An architectural description is often documented using an architecture template. 141

“architecture template”是用来指导和规范架构设计的通用模板，不是记录具体系统架构描述的工具. 实际的架构描述通常采用“architecture description language (ADL)”或特定的文档结构(如视图、模型等)来记录，而不是直接用模板.

(F) 3. 架构描述通常使用架构模板进行文档化.

( ) 13-4. An architectural decision is often documented using an architecture decision description template. 142

架构决策通常会用架构决策描述模板记录，有助于团队明确说明决策原因，包括背景、影响、备选方案等，方便后续沟通、维护和知识传递.

(T) 4. 架构决策通常使用架构决策描述模板进行文档化.

( ) 13-5. An architectural genre will often dictate the architectural approach that may used for the structure to be built. 143

一种架构类型(architectural genre)通常会决定所采用的架构方法(architectural approach). 在软件工程中，不同的架构类型(如客户端-服务器、分层架构、微服务等)有各自适合的设计和实现方式. 选择了某种架构类型，相关的架构方法和原则也会随之确定，用以指导系统结构设计.

(T) 5. 架构类型通常会决定用于构建结构的架构方法.

( ) 13-6. An architectural style encompasses which of the following elements? 144

A. constraints
B. set of components
C. semantic models
D. syntactic models

架构风格(architectural style)定义了系统中的组件类型、它们的关系和交互方式. 包括：A. constraints(约束)，规定组件之间的交互和连接限制；B. set of components(组件集合)，指明系统包含的基本构建模块；C. semantic models(语义模型)，描述各部分的意义和行为，帮助理解系统功能和数据流. D选项syntactic models(语法模型)主要关注组件间的结构关系，但不是架构风格的核心要素.

(ABC) 6. 架构风格包含以下哪些要素?
A. **约束**
B. **一组组件**
C. **语义模型**
D. 句法模型

( ) 13-7. To determine the architectural style or combination of styles that best fits the proposed system, requirements engineering is used to uncover 145

A. algorithmic complexity
B. characteristics and constraints
C. control and data
D. design patterns

需求工程(requirements engineering)在软件架构设计中，主要用来发现和明确系统需要具备的特性(characteristics)及必须遵守的约束条件(constraints)，这些信息对选择合适的架构风格或组合有直接影响. 算法复杂度、控制与数据、设计模式通常是后续设计或实现阶段关注的内容. 需求工程主要用于揭示系统的特性和约束.

(B) 7. 为了确定最适合拟议系统的体系结构风格或风格组合，需求工程用于发现
A. 算法复杂性
B. **特性和约束**
C. 控制和数据
D. 设计模式

( ) 13-8. Before an architectural pattern can be chosen for use in a specific system it must have a code implementation to facilitate its reuse. 146

架构模式是一种通用、可复用的解决方案，是抽象的设计理念，不依赖具体代码实现. 选择架构模式时，依据系统需求、质量属性和设计原则，而不是已有代码实现. 具体编码只在详细设计或实现阶段才需要，架构模式不需要事先有代码实现才能被选用.

(F) 8. 在为特定系统选择架构模式之前，必须有其代码实现以便于复用.

( ) 13-9. The criteria used to assess the quality of an architectural design should be based on system 147

A. accessibility
B. control
C. data
D. implementation

软件体系结构设计质量的评价标准主要包括系统的控制(B)和数据(C). 控制是指系统各部分之间的协调和交互机制，确保系统能正确管理流程和操作；数据涉及系统中数据的组织、流动和管理，保证数据的完整性和一致性. A选项的可访问性和D选项的实现细节主要在后续实现或用户体验阶段关注，不是体系结构设计评价的核心标准.

(BC) 9. 用于评估架构设计质量的标准应基于系统的
A. 可访问性
B. **控制**
C. **数据**
D. 实现

( ) 13-10. Software architectural considerations often interact with each other and moderate each other. 148

软件架构中的各种考虑因素(如性能、安全性、可扩展性、可维护性等)经常互相影响和制约. 例如，提高安全性可能影响性能，而优化性能又可能增加复杂性，影响可维护性. 因此，架构师在设计时需要权衡和协调这些不同因素，这些考虑不是孤立的，会相互作用、相互调节.

(T) 10. 软件架构方面的考虑因素通常会相互影响并相互调节.

( ) 13-11. Developer notes are not a reliable means of documenting architectural decisions 149

开发者笔记(Developer notes)可以有效记录架构决策，通常会详细说明技术选择的原因及相关权衡. 这些记录帮助团队成员理解系统架构的历史和设计意图，是可靠的文档手段之一.

(F) 11. 开发者笔记不是记录架构决策的可靠方式

( ) 13-12. During process of modeling the system in context, systems that interact with the target system are represented as 150

A. Peer-level systems
B. Subordinate systems
C. Superordinate systems
D. Working systems

系统建模中“系统环境(context)”是指在建模目标系统时，除了目标系统本身，还要考虑与其交互的其他系统. 常见的有三类：A. Peer-level systems(同级系统)，与目标系统处于同一层级并协作或通信；B. Subordinate systems(下级系统)，属于目标系统的子系统，被其控制或管理；C. Superordinate systems(上级系统)，即目标系统所在的更大系统或环境，相当于父系统. 这些系统都会在环境建模时被表示出来. D项不是标准分类.

(ABC) 12. 在对系统进行环境建模的过程中，与目标系统交互的系统被表示为
A. **同级系统**
B. **下级系统**
C. **上级系统**
D. 工作系统

( ) 13-13. Once selected, archetypes always need to be refined further as architectural design proceeds. 151

选定原型(archetypes)后，随着架构设计的推进，这些原型需要进一步细化. 原型是系统关键抽象的初步定义，在后续设计中，随着需求理解加深和系统结构具体化，原型通常要细化、扩展或调整，以更好满足实际需求. 原型不是固定不变的，后续设计中会不断完善和改进.

(T) 13. 一旦选定了原型，随着架构设计的推进，总是需要进一步细化原型.

( ) 13-14. Which of the following is not an example of infrastructure components that may need to be integrated into the software architecture? 152

A. Communications components
B. Database components
C. Interface components
D. Memory management components

基础设施组件包括通信、数据库、内存管理等，负责系统底层支持和资源管理. 界面组件用于与用户交互，属于应用层内容，不是基础设施组件.

(C) 14. 以下哪一项不是可能需要集成到软件架构中的基础设施组件的例子?
A. 通信组件
B. 数据库组件
C. **接口组件**
D. 内存管理组件

( ) 13-15. In the architecture trade-off analysis method the architectural style should be described using the 153

A. data flow view
B. module view
C. process view
D. user view

ATAM(架构权衡分析方法)中，架构风格通常用数据流视图、模块视图和进程视图来描述. 这些视图分别反映系统的数据流动、结构划分和运行时行为，便于全面分析架构的优缺点. 用户视图主要关注用户交互，不是ATAM描述架构风格的重点.

(ABC) 15. 在架构权衡分析方法中，架构风格应该使用以下哪种方式描述

A. **数据流视图**
B. **模块视图**
C. **进程视图**
D. 用户视图

( ) 13-16. A useful technique for evaluating the overall complexity of a proposed architecture is to look at the component 154

A. cohesion
B. flow dependencies
C. sharing dependencies
D. size

评估架构复杂性时，关键在于分析组件间的依赖关系. B(流程依赖)指组件间数据或控制流程的依赖, C(共享依赖)是指多个组件共享同一资源. 这两种依赖会增加系统复杂性, 是评估架构复杂性的常用方法. A(内聚)和D(大小)虽然影响设计，但对整体复杂性的直接评估作用不如B和C.

(BC) 16. 评估拟议架构整体复杂性的一个有用方法是查看组件的
A. 内聚性
B. **流程依赖**
C. **共享依赖**
D. 规模

( ) 13-17. Software architects need to create consensus among software team members and other stakeholders. 155

软件架构师需要在团队成员和其他利益相关者之间达成共识，因为他们不仅负责系统的整体设计，还要确保所有人对架构的理解和目标一致，这有助于项目顺利进行，减少沟通误差和后期修改成本. 因此，达成共识是架构师的重要工作之一.

(T) 17. 软件架构师需要在软件团队成员和其他利益相关者之间达成共识.

( ) 13-18. Pattern-based architectural reviews can be useful for project with short build cycles and volatile requirements. 156

基于模式的架构评审有助于团队快速识别和采用成熟设计方案，提高架构灵活性和可重用性. 对于需求变化频繁、开发周期紧张的项目，这种方法能加快决策、减少返工、提升开发效率，因此特别适合此类环境.

(T) 18. 基于模式的架构评审对于构建周期短且需求多变的项目是有用的.

( ) 13-19. Static architectural conformance checking assesses whether or not the source code matches the user visible requirements. 157

静态架构一致性检查(static architectural conformance checking)是检查源代码与架构设计(如模块结构、组件关系等)是否一致，并不直接判断源代码是否满足用户可见需求. 用户可见需求通常通过需求分析和动态测试来验证，而不是用静态架构检查.

(F) 19. 静态架构一致性检查评估源代码是否符合用户可见的需求.

( ) 13-20. Architectural design has no role in agile software process models. 158

虽然敏捷软件开发强调快速迭代和灵活应对变化，但架构设计在敏捷过程中依然重要. 敏捷方法采用"演进式架构"，在开发过程中逐步完善系统架构，并没有忽略架构设计. 良好的架构设计有助于系统的可维护性、扩展性和稳定性. 架构设计在敏捷流程中以更灵活、适应变化的方式进行.

(F) 20. 架构设计在敏捷软件过程模型中没有作用.

( ) 14-1. In the most general sense a component is a modular building block for computer software. 159

组件(component)是可以独立开发、测试、部署和维护的功能模块，它们像积木一样拼接成完整的软件系统. 模块化有助于提高系统的可维护性和可复用性，因此，组件是软件的模块化构建单元.

(T) 1. 在最广义的意义上，组件是计算机软件的模块化构建块.

( ) 14-2. In the context of object-oriented software engineering a component contains 160

A. attributes and operations
B. instances of each class
C. roles for each actor (device or user)
D. set of collaborating classes

在面向对象的软件工程中，组件是由一组协作的类组成，它们共同实现某个功能或业务需求，这些类通过接口进行通信和协作. A(属性和操作)描述的是类的内容，B(每个类的实例)指的是对象，C(每个参与者的角色)与用例建模相关，都不是组件的准确描述.

(D) 2. 在面向对象软件工程的背景下，一个组件包含
A. 属性和操作
B. 每个类的实例
C. 每个参与者(设备或用户)的角色
D. **一组协作的类**

( ) 14-3. In traditional software engineering modules must serve in which of the following roles? 161

A. Control component
B. Infrastructure component
C. Problem domain component
D. All of the above

在传统软件工程中，模块可承担多种角色，包括控制组件(实现流程控制)、基础结构组件(提供支持与服务)、问题域组件(直接实现业务需求). 模块可以是A、B或C中的任意一种.

(D) 3. 在传统软件工程中，模块必须扮演以下哪些角色?
A. 控制组件
B. 基础设施组件
C. 问题域组件
D. **以上全部**

( ) 14-4. Software engineers always need to cerate components from scratch in order to meet customer expectations fully. 162

题目说"软件工程师总是需要从零开始创建组件才能完全满足客户期望"，这是错误的. 实际上，软件工程师经常复用已有的组件、库或框架，这样可以提高开发效率、减少错误，同时也能满足客户需求. 只有在现有组件无法满足特定需求时，才需要从头开发，并不是所有情况下都要从零开始创建组件.

(F) 4. 软件工程师总是需要从头开始创建组件，以完全满足客户的期望.

( ) 14-5. Which of the following is not one of the four principles used to guide component-level design? 163

A. Dependency Inversion Principle
B. Interface Segregation Principle
C. Open-Closed Principle
D. Parsimonious Complexity Principle

A、B、C三项(依赖倒置原则、接口隔离原则、开闭原则)是常见的面向对象设计原则，也用于指导组件级设计. D项"Parsimonious Complexity Principle"(简约复杂性原则)不是四大原则之一，也不是软件工程中常见的设计原则名称，所以不属于组件级设计的四大原则.

(D) 5. 以下哪一项不是指导构件级设计的四项原则之一?
A. 依赖倒置原则
B. 接口隔离原则
C. 开闭原则
D. **简约复杂性原则**

( ) 14-6. The use of stereotypes can help identify the nature of components at the detailed design level. 164

在详细设计阶段，使用构造型(stereotype)可以区分和标识不同类型的组件，如控制类、实体类、边界类等. 这样有助于明确组件的职责和作用，提高设计的清晰度和可维护性，因此构造型有助于识别组件的性质.

(T) 6. 使用构件类型标记(stereotypes)可以帮助在详细设计层面识别构件的性质.

( ) 14-7. Classes and components that exhibit functional, layer, or communicational cohesion are relatively easy to implement, test, and maintain. 165

具有功能、层次或通信内聚的类和组件，因为内部各部分紧密相关，完成单一任务或相互协作，所以模块结构清晰、职责明确，出错时更容易定位和修复，也方便扩展和测试. 因此，这些类型的模块确实更容易实现、测试和维护.

(T) 7. 具有功能内聚、层次内聚或通信内聚的类和组件相对容易实现、测试和维护.

( ) 14-8. Software coupling is a sign of poor architectural design and can always be avoided in every system. 166

"软件耦合"是指系统中各模块间的依赖关系. 高耦合可能导致系统难以维护和扩展，但在实际设计中，完全避免耦合是不可能的，因为模块间需要交互. 良好架构应追求低耦合高内聚，但不是完全没有耦合. 耦合不是总能避免，也不一定代表设计差.

(F) 8. 软件耦合是糟糕架构设计的表现，并且在每个系统中总是可以避免.

( ) 14-9. In component design elaboration requires which of the following elements to be describe in detail? 167

A. Algorithms
B. Attributes
C. Interfaces
D. Operations

在构件设计细化阶段，需要详细描述属性(Attributes)、接口(Interfaces)和操作(Operations). 属性是构件的数据成员，接口描述构件与外部交互的方式，操作是构件可以执行的功能. 算法虽然重要，但通常在更具体的实现阶段详细设计，而不是构件设计细化时的重点.

(BCD) 9. 在构件设计细化中，以下哪些元素需要被详细描述?
A. 算法
B. **属性**
C. **接口**
D. **操作**

( ) 14-10. In component-level design persistent data sources refer to 168

A. Component libraries
B. Databases
C. Files
D. All of the above

在构件级设计中，persistent data sources(持久性数据源)指可以长期存储数据的信息源，如数据库(B)和文件(C). 它们在系统关闭后仍能保存数据. 组件库(A)主要用于复用代码，不属于持久数据源.

(BC) 10. 在构件级设计中，持久性数据源是指
A. 构件库
B. **数据库**
C. **文件**
D. 以上所有

( ) 14-11. WebApp content design at the component level focuses on content objects and the manner in which they interact. 169

WebApp内容设计在组件层级主要关注内容的组织、展示和结构，如文本、图片、链接等，而不是组件之间的交互. 组件层面的设计(如体系结构设计)才涉及组件的交互和协作，因此WebApp内容设计在组件层面并不主要关注内容对象之间的交互.

(F) 11. WebApp 在组件级别的内容设计关注内容对象及其交互方式.

( ) 14-12. A WebApp functional architecture describes the key functional components and how they interact with each other. 170

Web应用(WebApp)的功能架构描述了关键功能组件及它们之间的交互. 功能架构明确了Web应用的主要功能模块、各自的职责, 以及模块间的协作和通信方式, 有助于开发者理解系统结构, 合理分工和设计.

(T) 12. WebApp 功能架构描述了关键功能组件及其相互之间的交互方式.

( ) 14-13. Component-level design for mobile apps is not any different from component-based design for Web apps. 171

题目问移动应用的组件级设计和Web应用的组件级设计是否有区别. 组件级设计的核心思想是将系统分解为可重用、独立的模块(组件), 无论是移动应用还是Web应用, 这一原则和方法都是类似的. 两者都需要明确组件的接口、职责和交互方式, 所以在设计层面, 移动端和Web端的组件级设计没有本质区别.

(T) 13. 移动应用的构件级设计与Web应用的基于构件设计没有任何不同.

( ) 14-14. Which of these constructs is used in structured programming? 172

A. branching
B. condition
C. repetition
D. sequence

结构化程序设计包括三种基本结构: 顺序(sequence)、选择/分支(branching/condition)、循环(repetition). 条件(condition)是分支结构, 用于判断和选择; 循环(repetition)用于重复执行操作; 顺序(sequence)指代码依次执行. 选项A"branching"也是分支, 但标准答案用"condition"表示. B(condition)、C(repetition)、D(sequence)是结构化程序设计的基本构造.

(BCD) 14. 以下哪些结构用于结构化程序设计?
A. 分支
B. **条件**
C. **循环**
D. **顺序**

( ) 14-15. In component-based software engineering, the development team examines the requirements to see which are amenable to composition, rather than construction, before beginning detailed design tasks. 173

构件化软件工程(CBSE)的核心思想是, 开发团队在详细设计前会分析需求, 判断哪些部分可以用已有组件组合实现(复用), 而不是从头开发. 这样可以提高开发效率, 降低成本和风险. 题目描述符合CBSE的做法.

(T) 15. 在基于构件的软件工程中, 开发团队在开始详细设计任务之前, 会检查需求, 看看哪些需求适合通过组合实现, 而不是通过构建实现.

( ) 14-16. Which of the following is not one of the major activities of domain engineering? 174

A. analysis
B. construction
C. dissemination
D. validation

领域工程(domain engineering)的三大核心活动是分析(analysis)、构建(construction)、推广(dissemination). 分析是梳理和建模领域知识, 构建是开发可复用的领域资产, 推广是将这些资产应用到具体项目. 验证(validation)属于系统工程或软件开发过程中的活动, 不是领域工程的主要环节.

(D) 16. 下列哪一项不是领域工程的主要活动之一?
A. 分析
B. 构建
C. 传播
D. **验证**

( ) 14-17. Which of the following factors would not be considered during component qualification? 175

A. application programming interface (API)
B. development and integration tools required
C. exception handling
D. testing equipment required

组件选型时, 主要关注组件的接口(API)、异常处理能力和所需的开发及集成工具, 这些直接关系到组件能否集成进系统. 测试设备(testing equipment required)是测试阶段才会考虑的, 不属于组件自身特性, 因此不会在组件选型时考虑.

(D) 17. 下列哪些因素在组件资格认证过程中不会被考虑?
A. 应用程序编程接口(API)
B. 所需的开发和集成工具
C. 异常处理
D. **所需的测试设备**

( ) 14-18. Which is the following is a technique used for component wrapping? 176

A. black-box wrapping
B. clear-box wrapping
C. gray-box wrapping
D. white-box wrapping

题目涉及组件封装(component wrapping)的方法. B选项clear-box wrapping并不是常见术语, 标准术语有black-box(黑盒)、white-box(白盒)、gray-box(灰盒)封装. 实际软件工程中, component wrapping常用技术是black-box wrapping(黑盒封装), 即不需要了解组件内部实现, 只通过接口交互. 题目给出的答案是B, clear-box wrapping不是标准封装技术. 常见封装技术有black-box、gray-box、white-box, 正确答案应为A.

(B) 18. 下列哪一项是用于组件封装的技术?
A. 黑盒封装
B. **明盒封装**
C. 灰盒封装
D. 白盒封装

( ) 14-19. Which of the following is not one of the issues that form a basis for design for reuse? 177

A. object-oriented programming
B. program templates
C. standard data
D. standard interface protocols

B(程序模板)、C(标准数据)和D(标准接口协议)都是支持软件复用的重要机制, 提供了可重复利用的结构、数据和接口. A(面向对象程序设计)是一种编程范式, 有助于提高复用性, 但本身不是设计复用的具体基础要素, 因此不属于复用设计基础问题.

(A) 19. 下列哪一项不是构成可复用性设计基础的问题之一?
A. **面向对象编程**
B. 程序模板
C. 标准数据
D. 标准接口协议

( ) 14-20. In a reuse environment, library queries are often characterized using the _____ element of the 3C Model. 178

A. concept
B. content
C. context
D. all of the above

3C模型包括concept(概念)、content(内容)和context(背景/上下文). 在软件复用时, 库查询会用concept描述查询主题或功能, 用content指定所需资源的内容, 用context说明使用场景和条件. 因此, 库查询通常涉及3C模型的全部要素.

(D) 20. 在复用环境中, 库查询通常使用3C模型中的_____要素来描述.
A. 概念
B. 内容
C. 上下文
D. **以上全部**

( ) 15-1. Which of the following interface design principles does not allow the user to remain in control of the interaction with a computer? 179

A. allow interaction to interruptible
B. allow interaction to be undoable
C. hide technical internals from casual users
D. only provide one rigidly defined method for accomplishing a task

A、B选项(允许中断和撤销操作)让用户可以灵活掌控操作过程. C选项(隐藏技术细节)是为了简化用户体验, 不会限制用户的控制. D选项(只提供一种固定的方法完成任务)限制了用户的选择权, 用户不能根据自己的习惯或需求操作, 因此不允许用户保持对交互的控制.

(D) 1. 下列哪一项界面设计原则不允许用户控制与计算机的交互?
A. 允许交互可被中断
B. 允许交互可撤销
C. 对普通用户隐藏技术细节
D. **只提供一种严格定义的方法来完成任务**

( ) 15-2. Which of the following interface design principles reduce the user's memory load? 180

A. define intuitive shortcuts
B. disclose information in a progressive fashion
C. establish meaningful defaults
D. provide an on-line tutorial

A. 定义直观的快捷方式(define intuitive shortcuts)让用户能快速操作, 减少记忆繁琐步骤的需求. B. 以渐进方式披露信息(disclose information in a progressive fashion)只在需要时展示相关内容, 避免用户一次记住太多信息. C. 建立有意义的默认值(establish meaningful defaults)系统自动给出合理默认设置, 用户不必每次都记住和输入, 减轻记忆负担. D. 提供在线教程(provide an on-line tutorial)虽能帮助学习, 但不直接减少操作过程中的记忆负担. A、B、C这些原则都直接帮助用户减少需要记住的信息量.

(ABC) 2. 下列哪些界面设计原则可以减少用户的记忆负担?
A. **定义直观的快捷方式**
B. **以渐进的方式披露信息**
C. **设定有意义的默认值**
D. 提供在线教程

( ) 15-3. The reason for reducing the user's memory load is make his or her interaction with the computer quicker to complete. 181

减少用户的记忆负担(memory load)主要是为了让用户更容易、更顺畅地使用系统, 减少出错、提升使用体验, 而不仅仅是为了加快操作速度. 虽然降低记忆负担有时会提高效率, 但核心目的是提升可用性和易用性, 而不是单纯追求操作速度.

(F) 3. 减少用户记忆负担的原因是使其与计算机的交互能够更快完成.

( ) 15-4. Interface consistency implies that 182

A. each application should have its own distinctive look and feel
B. input mechanisms remain the same throughout the application
C. navigational methods are context sensitive
D. visual information is organized according to a design standard

界面一致性(interface consistency)是指软件界面在交互方式和视觉布局上保持统一. B选项"输入机制在整个应用中保持一致"体现了用户在使用应用时输入方式不会因界面不同而变化, 有助于用户学习和操作. D选项"视觉信息根据设计标准进行组织"强调了视觉风格和布局的一致性, 也提升了用户体验. A和C分别强调个性化和上下文敏感, 不符合界面一致性的要求.

(BD) 4. 接口一致性意味着
A. 每个应用程序都应该有自己独特的外观和风格
B. **输入机制在整个应用程序中保持一致**
C. 导航方法依赖于上下文
D. **视觉信息按照设计标准进行组织**

( ) 15-5. If past interactive models have created certain user expectations it is not generally good to make changes to the model. 183

用户习惯和用户体验很重要. 如果以前的交互模型让用户形成了固定的使用习惯和期望, 随意改变模型会让用户困惑、不适应, 还可能影响软件的可用性和满意度. 因此, 不建议随意改变已经建立的交互模型.

(T) 5. 如果以往的交互模型已经让用户形成了某些期望, 通常不宜对该模型进行更改.

( ) 15-6. Which model depicts the profile of the end users of a computer system? 184

A. design model
B. implementation model
C. user model
D. user's model

题目问哪个模型描述了计算机系统终端用户的特征. C项user model是用来描述终端用户的背景、需求和行为特征, 帮助设计和开发人员了解用户群体. design model和implementation model主要关注系统设计和实现细节, 不涉及用户特征. user's model指的是用户自己对系统的理解, 不是开发者用来描述用户特征的模型.

(C) 6. 哪种模型描述了计算机系统最终用户的特征?
A. 设计模型
B. 实现模型
C. **用户模型**
D. 用户的模型

( ) 15-7. Which model depicts the image of a system that an end user creates in his or her head? 185

A. design model
B. user model
C. system model
D. system perception

考查用户对系统的理解和感知. system perception(系统感知)指的是终端用户在头脑中形成的对系统的整体印象和理解, 也就是用户认为系统是什么、怎么工作的, 这种印象不一定和实际系统完全一致. design model(设计模型)、system model(系统模型)和 user model(用户模型)都是开发者在开发过程中建立的模型, 不是用户自发形成的.

(D) 7. 哪种模型描述了最终用户在其头脑中创建的系统形象?
A. 设计模型
B. 用户模型
C. 系统模型
D. **系统感知**

( ) 15-8. Which model depicts the look and feel of the user interface along with all supporting information? 186

A. implementation model
B. user model
C. user's model
D. system perception

implementation model(实现模型)描述系统的具体实现, 包括用户界面的外观与感觉(look and feel)、操作方式和支持信息. 它帮助开发人员准确实现用户界面, 确保系统最终外观与用户期望一致. 其他选项虽然涉及用户和系统的观点, 但只有A选项关注界面实现的具体细节.

(A) 8. 哪种模型描述了用户界面的外观和感觉以及所有支持信息?
A. **实现模型**
B. 用户模型
C. 用户的模型
D. 系统感知

( ) 15-9. Which of these framework activities is not normally associated with the user interface design processes? 187

A. cost estimation
B. interface construction
C. interface validation
D. user and task analysis

用户界面设计主要包括用户和任务分析(理解用户需求和使用场景)、界面构建(设计和实现界面)、界面验证(测试和评估界面). 成本估算(cost estimation)属于项目管理活动, 通常不包括在用户界面设计流程中.

(A) 9. 下列哪些框架活动通常不与用户界面设计过程相关?
A. **成本估算**
B. 界面构建
C. 界面验证
D. 用户与任务分析

( ) 15-10. Which approach(es) to user task analysis can be useful in user interface design? 188

A. have users indicate their preferences on questionnaires
B. rely on the judgement of experienced programmers
C. study existing computer-based solutions
D. observe users performing tasks manually

用户任务分析在用户界面设计中常用的方法有研究已有的计算机化解决方案(C)和观察用户手工完成任务(D). 研究现有系统可以了解现有界面设计的优缺点和改进空间; 观察用户实际操作能直接发现他们的工作流程、需求和遇到的问题, 有助于设计更符合实际需求的界面. A选项"让用户在问卷上表达偏好"只能获得主观感受, 无法深入分析具体任务过程. B选项"依赖有经验程序员的判断"则容易带有设计者偏见, 不能真实反映用户的实际任务.

(CD) 10. 哪些用户任务分析方法在用户界面设计中是有用的?
A. 让用户在问卷上标明他们的偏好
B. 依赖有经验程序员的判断
C. **研究现有的基于计算机的解决方案**
D. **观察用户手工执行任务**

( ) 15-11. Object-oriented analysis techniques can be used to identify and refine user task objects and actions without any need to refer to the user voice. 189

面向对象分析技术需要参考用户需求和实际操作习惯, 这是分析对象和任务的重要依据. 如果不考虑用户意见和需求, 识别出的对象和任务可能无法满足实际需求, 因此用户声音在对象识别和任务分析中非常重要.

(F) 11. 面向对象分析技术可以用来识别和细化用户任务对象和操作, 而无需参考用户的声音.

( ) 15-12. The computer's display capabilities are the primary determinant of the order in which user interface design activities are completed. 190

题目说电脑显示能力是决定用户界面设计活动顺序的主要因素, 实际上, 用户界面设计的顺序主要取决于用户需求、任务逻辑和易用性等, 而不是硬件的显示能力. 显示能力只会影响设计细节, 但不是主要决定因素, 因此这个说法不正确.

(F) 12. 计算机的显示能力是决定用户界面设计活动完成顺序的主要因素.

( ) 15-13. It is sometimes possible that the interface designer is constrained by environmental factors that mitigate against ease of use for many users. 191

界面设计师有时会受到环境因素限制, 这些限制可能影响用户的易用性. 在实际开发中, 设计师需要考虑硬件、操作环境、法规等外部条件, 这些因素有时会妨碍界面的友好性和易用性, 这种情况确实存在.

(T) 13. 有时界面设计者会受到环境因素的限制, 这些因素会影响许多用户的易用性.

( ) 15-14. One means of defining user interface objects and actions is to conduct a grammatical parse of the user scenario. 192

定义用户界面对象和动作的一种方法, 是对用户场景进行语法分析. 通过分析用户场景描述中的名词和动词, 可以识别界面中的对象(如"文件"、"按钮"等名词)和动作(如"打开"、"保存"等动词). 这种方法有助于系统地抽取用户界面应包含的元素和功能, 是界面设计中常用的分析方法.

(T) 14. 定义用户界面对象和操作的一种方法是对用户场景进行语法分析.

( ) 15-15. Interface design patterns typically include a complete component-level design (design classes, attributes, operations, and interfaces). 193

界面设计模式主要关注组件间的交互和通信, 不包括完整的组件级设计, 如详细的类、属性、操作等内容. 完整的组件级设计属于更详细的系统设计阶段, 而设计模式只提供结构或交互的通用解决方案, 不涉及实现细节.

(T) 15. 接口设计模式通常包括完整的组件级设计(设计类、属性、操作和接口).

( ) 15-16. Several common design issues surface for almost every user interface including 194

A. adaptive user profiles
B. error handling
C. resolution of graphics displays
D. system response time

B(错误处理)和D(系统响应时间)是几乎每个用户界面都会遇到的重要设计问题. 错误处理涉及用户操作失误时系统的反馈和修正能力, 系统响应时间影响用户操作的流畅性和体验. A(自适应用户画像)和C(图形显示分辨率)虽然与界面设计有关, 但不是所有用户界面都要普遍考虑的问题.

(BD) 16. 并非每一个用户界面都会遇到一些常见的设计问题, 包括
A. 自适应用户配置文件
B. **错误处理**
C. 图形显示分辨率
D. **系统响应时间**

( ) 15-17. It is more important to capture the user's attention with flashy features than ergonomically sound screen layouts when building a WebApp. 195

在构建Web应用时, 人体工学合理、用户友好的界面布局比花哨的功能更重要. 良好的布局提升用户体验, 使操作更高效、更舒适, 过多追求炫目效果可能分散注意力, 降低易用性. 设计时应优先考虑界面的实用性和易用性.

(F) 17. 在构建Web应用时, 用炫目的功能吸引用户注意力比符合人体工学的界面布局更重要.

( ) 15-18. Several usability measures can be collected while observing users interacting with a computer system including 196

A. down time for the application
B. number of user errors
C. software reliability
D. time spent looking at help materials

可用性度量(usability measures)是指在观察用户与计算机系统交互时, 用于衡量系统易用性的数据. B选项"用户错误的次数"(number of user errors)直接反映系统的易用性, 错误越多说明系统越难用. D选项"查阅帮助材料的时间"(time spent looking at help materials)反映用户是否容易理解和使用系统,

查帮助时间越长, 说明可用性可能较差. A选项"应用程序的宕机时间"和C选项"软件可靠性"属于系统可靠性和稳定性指标, 不是可用性的直接度量.

(BD) 18. 在观察用户与计算机系统交互时, 可以收集以下几种可用性度量:
A. 应用程序的停机时间
B. **用户错误的数量**
C. 软件可靠性
D. **用户查阅帮助材料所花的时间**

( ) 16-1. Which of the following is not one of the elements of a design pattern? 197

A. context
B. environment
C. problem
D. solution

设计模式有三个主要要素: 上下文(context)、问题(problem)和解决方案(solution). 它们分别说明适用场景、需要解决的问题以及具体的解决方法. "environment"不是设计模式的标准组成部分.

(B) 1. 下列哪一项不是设计模式的要素?
A. 上下文
B. **环境**
C. 问题
D. 解决方案

( ) 16-2. RubberNecking is an example of a classic generative pattern. 198

RubberNecking 不是经典的生成型(generative)模式. 生成型模式指可以复用、推广, 帮助开发者在不同场景下生成解决方案的模式, 如MVC、工厂方法等. RubberNecking 并不是这类被广泛认可和推广的经典生成型模式.

(F) 2. RubberNecking 是一个经典生成型模式的例子.

( ) 16-3. A frame work is a reusable mini-architecture that serves as a foundation which other design patterns can be applied? 199

框架(framework)是一种可复用的小型架构, 提供通用结构和基础组件, 开发者可以在此基础上扩展和定制, 并结合各种设计模式解决具体问题. 框架比单一设计模式大, 但比完整应用系统小, 是连接设计模式和具体应用的桥梁.

(T) 3. 框架是一种可重用的微型架构, 它作为基础, 可以应用其他设计模式.

( ) 16-4. Finding patterns built by others that address design problems is often more difficult that recognizing patterns in the application to be built. 200

寻找他人已建立、用于解决设计问题的模式, 通常比在自己要开发的应用中识别模式更困难. 原因在于, 查找已有设计模式需要理解大量模式、分析它们的适用条件, 并判断是否适合当前问题. 而在自己应用中识别模式, 通常基于对自身系统和业务的深刻理解, 相对更直接、更容易. 因此, 寻找外部模式的难度通常更大.

(T) 4. 寻找他人构建的、用于解决设计问题的模式, 通常比识别待开发应用中的模式更困难.

( ) 16-5. A pattern language 201

A. encompasses a collection of patterns
B. is implemented using hypertext
C. resembles the structure of natural languages
D. None of the above

pattern language(模式语言)是指一组经过整理和组织的设计模式的集合, 用于解决某一领域中的常见问题. 它不仅包含单个模式, 还强调多个模式之间的有机联系. B只是实现方式之一, C只是比喻, 并不是定义.

(A) 5. 一个模式语言
A. **包含一组模式**
B. 使用超文本实现
C. 类似于自然语言的结构
D. 以上都不是

( ) 16-6. The concepts and techniques discussed for _____ can be used in the conjunction with a pattern-based approach. 202

A. Architectural design
B. Component-level design
C. User interface design
D. All of the above

模式(pattern)是一种可复用的设计方案, 用来解决常见的软件设计问题. 在架构设计、构件级设计和用户界面设计中, 都可以结合使用模式来规范和优化设计. 模式相关的概念和技术可以在这三个层次都被应用.

(D) 6. 讨论过的_____的概念和技术可以与基于模式的方法结合使用.
A. 架构设计
B. 构件级设计
C. 用户界面设计
D. **以上所有**

( ) 16-7. It is important to reduce the coupling among design patterns so that they can be treated as independent entities. 203

设计模式的核心思想是解决特定设计问题, 许多设计模式之间存在一定关联. 过度追求完全独立的设计模式会丧失它们协同解决复杂问题的优势. 在实际软件设计中, 适当的耦合可以让系统结构更灵活、可扩展, 因此不需要刻意减少设计模式之间的耦合.

(F) 7. 降低设计模式之间的耦合性很重要, 这样它们可以被视为独立的实体.

( ) 16-8. Real life design solutions may not always lend themselves to a top-down approach. 204

现实中的设计解决方案不一定总能采用自顶向下的方法. 因为在实际软件开发中, 需求不明确、技术限制或现有系统约束等原因, 常常不能完全按照自顶向下(从总体到细节分解)来设计, 很多时候还需要结合自底向上或混合方法. 所以, 现实中的设计不总是适合单一的自顶向下方法.

(T) 8. 现实生活中的设计解决方案可能并不总是适合自顶向下的方法.

( ) 16-9. Which of the following problem types are used to label columns in a pattern organizing table? 205

A. Business
B. Context
C. Database
D. Infrastructure

模式组织表用于对设计模式进行分类和描述, 表格的列通常标注与实现相关的技术领域, 如数据库(Database)和基础设施(Infrastructure), 这些属于实际开发中需要考虑的具体实现问题. Business(业务)和Context(上下文)主要描述应用场景或业务需求, 不属于技术实现的问题类型.

(CD) 9. 下列哪些问题类型用于在模式组织表中标注列?
A. 业务
B. 上下文
C. **数据库**
D. **基础设施**

( ) 16-10. Most mistakes in pattern-based design can be avoided by judicious use of review techniques. 206

题目说大多数基于模式的设计中的错误可以通过合理使用评审技术来避免. 评审(如代码评审、设计评审)是软件工程中发现和纠正设计过程中错误和不足的有效方法. 尤其是在使用设计模式时, 团队成员通过评审能发现模式应用不当、理解有误或实现不规范等问题, 从而减少设计中的错误. 合理利用评审技术确实能避免许多基于模式设计的错误.

(T) 10. 通过明智地使用评审技术, 大多数基于模式的设计错误是可以避免的.

( ) 16-11. Before choosing an architectural design pattern it must be assessed for its appropriateness to the application and overall architectural style. 207

在选择架构设计模式前, 必须评估其是否适合具体应用和整体架构风格. 每种架构模式都有特定优缺点, 只有与实际需求和系统架构风格相匹配, 才能发挥最佳效果. 盲目选择可能导致系统难以维护或无法满足需求, 所以评估适用性是重要步骤.

(T) 11. 在选择架构设计模式之前, 必须评估其对应用程序和整体架构风格的适用性.

( ) 16-12. Unlike architectural patterns, component-level design patterns may be applied to solve subproblems without regard to system context. 208

无论是架构层次还是组件层次的设计模式, 都需要考虑系统的上下文(即系统的具体环境和需求). 组件级设计模式也必须确保适合当前子问题和系统整体架构环境, 否则可能导致设计不协调或出现新问题. 因此, 不能认为组件级模式可以不考虑系统上下文直接应用.

(F) 12. 与架构模式不同, 组件级设计模式可以在不考虑系统上下文的情况下应用于解决子问题.

( ) 16-13. Most user interface design patterns fall with in one of _____ categories of patterns. 209

A. 5
B. 10
C. 25
D. 100

考查用户界面(UI)设计模式的分类. 大多数用户界面设计模式通常被归纳为10个主要类别, 如导航、输入、反馈等. 这种分类便于设计师在开发中查找和应用合适的模式.

(B) 13. 大多数用户界面设计模式属于以下几类模式之一.
A. 5
B. **10**
C. 25
D. 100

( ) 16-14. WebApp design patterns can be classified by considering which of the dimensions listed below? 210

A. Aesthetics
B. Design focus
C. Granularity
D. Usability

WebApp设计模式的分类通常基于"设计关注点"(Design focus)和"粒度"(Granularity). "设计关注点"指模式主要解决的问题领域, 如导航、内容组织等; "粒度"指模式的应用范围, 是针对整个系统还是某个具体功能. Aesthetics(美学)和Usability(可用性)更多是设计目标或评估标准, 而不是分类依据.

(BC) 14. WebApp设计模式可以通过考虑下列哪些维度进行分类?
A. 美学
B. **设计重点**
C. **粒度**
D. 可用性

( ) 16-15. Which of the following are levels of design focus that can be used to categorize WebApp patterns? 211

A. Behavioral patterns
B. Functional patterns
C. Layout patterns
D. Navigation patterns

WebApp(网页应用)模式的设计层次包括功能层(Functional patterns)和导航层(Navigation patterns). 功能模式关注系统实现的功能, 导航模式关注用户在系统中的移动和信息查找. 行为模式(Behavioral patterns)和布局模式(Layout patterns)虽然重要, 但不是WebApp设计层次的主要分类.

(BD) 15. 以下哪些是可用于对WebApp模式进行分类的设计关注层次?

A. 行为模式
B. **功能模式**
C. 布局模式
D. **导航模式**

( ) 16-16. Which of the levels of granularity that can be used to describe WebApp patterns? 212

A. Architectural patterns
B. Component patterns
C. Design patterns
D. Interactions patterns

本题考查Web应用(WebApp)模式描述时的粒度层次. 粒度指描述的详细程度. A. Architectural patterns(架构模式)是最高层次, 描述整个系统结构. B. Component patterns(组件模式)描述系统的某个部分或模块. C. Design patterns(设计模式)关注具体设计细节, 如类、对象之间的关系. 这三种都是常见的粒度层次. D. Interaction patterns(交互模式)主要指用户界面层面的人机交互, 不属于WebApp结构和实现的常用模式粒度范围.

(ABC) 16. 哪些粒度级别可以用来描述Web应用程序的模式?
A. **架构模式**
B. **组件模式**
C. **设计模式**
D. 交互模式

( ) 16-17. Mobile app user interface patterns can be represented as a collection of best of breed screen images. 213

移动应用界面模式可以用一组优秀的屏幕图像来表示, 指的是常见且效果好的界面设计方案(如登录页、列表页等)可以通过收集这些界面截图来展示和总结, 方便后续设计参考和复用. 这在软件工程实际工作中很常见, 有助于标准化和提升用户体验.

(T) 17. 移动应用用户界面模式可以表示为一组最佳实践的屏幕图像集合.

( ) 17-1. Which of the following characteristics should not be used to assess the quality of a WebApp? 214

A. aesthetics
B. reliability
C. maintainability
D. usability

aesthetics(美观)虽然影响用户体验, 但主要关注界面设计和视觉效果, 不属于软件工程中衡量软件质量的核心属性. reliability(可靠性)、maintainability(可维护性)、usability(可用性)都是软件工程衡量WebApp质量的重要标准. A aesthetics不是评估WebApp质量的基本特性.

(A) 1. 下列哪项特性不应用于评估Web应用程序的质量?
A. **美观性**
B. 可靠性
C. 可维护性
D. 可用性

( ) 17-2. Which of the following are design goals for every WebApp? 215

A. Simplicity
B. Consistency
C. Navigability
D. Visual appeal

Web应用程序设计的基本目标包括: 简洁(Simplicity), 方便用户理解和使用; 一致性(Consistency), 界面和操作风格统一; 可导航性(Navigability), 便于用户在功能和页面间切换; 视觉吸引力(Visual appeal), 页面美观, 提升用户体验. 这些都是高质量WebApp必须具备的设计目标.

(ABCD) 2. 以下哪些是每个Web应用程序的设计目标?
A. **简洁性**
B. **一致性**
C. **可导航性**
D. **视觉吸引力**

( ) 17-3. Which of the following not part of the design pyramid for WebE design? 216

A. Architectural design
B. Business case design
C. Content design
D. Navigation design

WebE(Web工程)设计金字塔包括: 体系结构设计(Architectural design)、内容设计(Content design)、导航设计(Navigation design), 这些都直接关系到网站结构和用户体验. Business case design(商业案例设计)关注的是商业价值和可行性分析, 不属于WebE设计金字塔的组成部分.

(B) 3. 以下哪项不是WebE设计金字塔的一部分?
A. 架构设计
B. **业务案例设计**
C. 内容设计
D. 导航设计

( ) 17-4. With WebApps content is everything, a poorly defined user interface will be quickly overlooked by frequent users. 217

内容固然重要, 但在Web应用中, 用户界面同样关键. 界面设计不好会影响用户体验, 即使内容很好, 用户也可能不愿继续使用. 因此, 优秀的Web应用需要优质内容和良好的用户界面.

(F) 4. 对于Web应用程序, 内容就是一切, 频繁用户会很快忽略一个界面设计不佳的用户界面.

( ) 17-5. Which of these are WebApp interaction mechanisms? 218

A. Graphic icons
B. Graphic images
C. Navigation menus
D. All of the above

交互机制是指用户与Web应用进行交互的方式. 图形图标、图形图像和导航菜单, 都是用户可以点击或操作的界面元素, 帮助用户和系统进行信息交流和功能操作. 这三项都是WebApp常见的交互机制.

(D) 5. 以下哪些是Web应用的交互机制?
A. 图形图标
B. 图形图像
C. 导航菜单
D. **以上全部**

( ) 17-6. Screen layout design has several widely accepted standards based on human factors research. 219

虽然屏幕布局设计受人因工程(人机交互、用户体验等)研究影响, 但实际上没有几个被广泛接受的统一标准. 通常是一些设计原则和最佳实践, 如一致性、简洁性、可用性等, 而不是明确、行业公认的详细标准. 不同组织和界面类型可能采用不同的指导规范.

(F) 6. 屏幕布局设计有几个基于人因研究的广泛认可的标准.

( ) 17-7. Graphic design considers every aspect of the look and feel of a WebApp. 220

图形设计不仅关注界面的颜色和图片, 还包括布局、字体、按钮样式、交互效果等所有影响用户视觉和操作体验的元素. 其目标是让WebApp在视觉上吸引用户, 使用时舒适、易懂. 因此, 图形设计涵盖了WebApp外观和感觉的各个方面.

(T) 7. 图形设计考虑了Web应用外观和体验的每一个方面.

( ) 17-8. Content design is conducted by 221

A. Copywriters and graphic designer
B. Web engineers
C. both a and b
D. none of the above

内容设计的参与者不仅有文案人员(copywriters)和图形设计师(graphic designer), 网页工程师(web engineers)也常常参与, 因为他们负责将内容实现为可用的网页或系统. 实际开发中, 内容设计通常是多角色协作完成的, 所以A和B都包括.

(C) 8. 内容设计由谁进行
A. 文案撰写者和图形设计师
B. 网页工程师
C. **a 和 b 都是**
D. 以上都不是

( ) 17-9. Content objects have both information attributes defined during analysis and implementation specific attributes specified during design. 222

内容对象(Content Object)在分析阶段定义与业务相关的信息属性, 如数据的基本内容和结构; 在设计阶段补充实现相关的属性, 如数据库主键、存储格式等. 因此, 内容对象包含分析阶段的信息属性和设计阶段的实现相关属性.

(T) 9. 内容对象既具有在分析阶段定义的信息属性, 也具有在设计阶段指定的实现特定属性.

( ) 17-10. Content objects are not normally chunked into Web pages until the implementation activities begin. 223

内容对象在Web工程的设计阶段就已经被合理划分和组织到各网页中, 而不是等到实现阶段才进行. 这样能保证网站结构清晰、用户体验良好, 并为实现阶段提供明确指导. 因此, 题目中的说法不正确.

(F) 10. 内容对象通常不会在实现活动开始之前被分块到网页中.

( ) 17-11. Content architecture and WebApp architecture are pretty much the same thing for many WebApps? 224

Content architecture关注网站内容的组织、结构和呈现, 如页面、导航、信息分类等; WebApp architecture关注Web应用的技术结构, 包括前端、后端、数据库、服务器等. 两者关注点不同, 不能等同.

(F) 11. 对于许多Web应用来说, 内容架构和Web应用架构几乎是同一回事吗?

( ) 17-12. Which of the following is not one of the content architectural structures used by web engineers? 225

A. linear
B. grid
C. hierarchical
D. parallel

Web工程常见的内容架构结构有线性(linear)、网格(grid)和层次(hierarchical), 分别对应内容按顺序、按二维网格和按层级组织. "parallel(并行)"不是Web工程中常用的内容架构结构.

(D) 12. 以下哪一项不是Web工程师使用的内容架构结构?
A. 线性
B. 网格
C. 层次结构
D. **并行**

( ) 17-13. MVC is a three layer architecture that contains a 226

A. machine, view, content objects
B. model, view, and content objects
C. model, view, and controller
D. machine, view, controller

MVC是常见的软件架构模式, 全称为Model-View-Controller. 它将应用分为Model(负责数据和业务逻辑)、View(负责界面显示)、Controller(负责协调模型和视图的交互). 这种分层有助于代码结构清晰、便于维护和扩展.

(C) 13. MVC 是一种包含三层结构的架构，它包括
A. 机器、视图、内容对象
B. 模型、视图、内容对象
C. **模型、视图、控制器**
D. 机器、视图、控制器

( ) 17-14. Web navigational design involves creating a semantic navigational unit for each goal associated with each defined user role. 227

Web导航设计通常会根据不同用户角色(如普通用户、管理员等)及其目标(如浏览信息、提交表单等)，为每个目标设计清晰、易理解的导航单元(如菜单、链接等)，确保用户顺利实现各自目标．这有助于提升用户体验和系统可用性．
(T) 14. Web导航设计涉及为与每个已定义用户角色相关的每个目标创建一个语义导航单元.

( ) 17-15. To allow the user to feel in control of a WebApp, it is a good idea to mix both horizontal and vertical navigation mechanisms on the same page. 228

在同一页面上混合水平和垂直导航会让用户困惑，降低控制感和易用性．WebApp设计应保持导航方式一致和简单，让用户容易理解和操作，从而让用户真正感到在控制应用.
(F) 15. 为了让用户感觉能够控制Web应用，在同一页面上混合使用水平和垂直导航机制是一个好主意.

( ) 17-16. Component level design for WebApps is very similar to component level design for other software delivery environments. 229

Web应用(WebApps)的组件级设计与其他软件开发环境的组件级设计非常相似．无论是Web应用还是桌面应用，组件级设计的核心都是将系统划分为可重用、可维护的模块，这些模块有明确的接口、功能和内部实现．虽然Web应用有特定技术(如前后端分离、Web框架等)，但设计原则和过程基本一致，比如高内聚、低耦合、信息隐藏等．因此二者非常相似.
(T) 16. Web应用的构件级设计与其他软件交付环境的构件级设计非常相似.

( ) 17-17. Which of these is not one of the design activities associated with object-oriented hypermedia design? 230

A. abstract interface design
B. conceptual design
C. content design
D. navigational design

面向对象超媒体设计(Object-Oriented Hypermedia Design，OOHD)包括概念设计(conceptual design)、导航设计(navigational design)和抽象界面设计(abstract interface design)，分别关注系统结构、用户浏览方式和界面元素的抽象．内容设计(content design)不是OOHD特有的设计活动，而是更常见于一般信息系统或网站设计.
(C) 17. 以下哪一项不是面向对象超媒体设计相关的设计活动?
A. 抽象界面设计
B. 概念设计
C. **内容设计**
D. 导航设计

( ) 17-18. UML does not have any representation schemas that are useful in building WebApp design models. 231

UML(统一建模语言)提供了多种图示，如用例图、类图、顺序图、活动图等，这些都是Web应用设计中常用的建模工具，能帮助分析和设计WebApp的结构和行为．因此，UML有助于WebApp设计建模.
(F) 18. UML 没有任何对构建 WebApp 设计模型有用的表示方案.

( ) 18-1. MobileApps must be designed take intermittent connectivity outages. 232

移动应用在设计时要考虑到间歇性网络连接中断．移动设备常常会遇到信号弱或断网的情况，所以软件工程师需要保证应用在网络不稳定或断开的情况下也能正常工作，比如通过本地缓存、断点续传等机制．这是移动应用开发中的一个重要设计考虑.
(T) 1. 移动应用程序的设计必须考虑间歇性连接中断.

( ) 18-2. Modern electronics allow developers to ignore the power demands made by a MobileApp. 233

现代电子技术并不允许开发者忽略移动应用的功耗需求．虽然移动设备硬件性能提升、电池技术进步，但移动应用的能耗依然是开发时必须考虑的重要因素．手机电池容量有限，应用耗电过大会影响用户体验，导致电量快速消耗，甚至被用户卸载．因此，开发者需要优化应用，合理使用资源，降低能耗.
(F) 2. 现代电子技术允许开发者忽略移动应用的电源需求.

( ) 18-3. A MobileApp is assessed for usability and accessibility before beginning the next increment begins. 234

在进入下一个增量开发阶段前，需要对移动应用进行可用性(用户是否容易使用)和可访问性(不同用户是否都能方便访问)评估．增量开发模型要求每完成一个增量，都要进行评估和测试，以便及时发现和修正问题，保证软件质量，为下一个增量打好基础．因此，每个增量开发完成后、下一个增量开始前，都要进行这样的评估.
(T) 3. 在开始下一个增量之前，需要对移动应用程序进行可用性和可访问性评估.

( ) 18-4. Which of the following characteristics should not be used to assess the quality of a MobileApp? 235

A. aesthetics
B. reliability
C. maintainability
D. usability

B、C、D(reliability 可靠性、maintainability 可维护性、usability 易用性)都是软件工程核心质量属性，直接影响软件表现和用户体验．A项 aesthetics(美观性/美学)虽然影响用户吸引力，但不是软件工程正式定义的质量评估标准，因此不应作为主要依据.
(A) 4. 下列哪些特性不应被用来评估移动应用的质量?
A. **美观性**
B. 可靠性
C. 可维护性
D. 可用性

( ) 18-5. Quality function deployment is not necessary when implementing MobileApp user stories? 236

质量功能展开(QFD)是一种将用户需求转化为技术要求的方法，有助于团队理解并满足用户需求．在开发移动应用用户故事时，应用QFD可以确保开发的功能符合用户期望，提高软件质量和用户满意度．因此，不能认为QFD是不必要的.
(F) 5. 在实现移动应用用户故事时，质量功能展开(QFD)不是必需的吗?

( ) 18-6. Using highly adaptive contextual interfaces is a good way to deal with device limitations like screen size. 237

高度自适应的上下文界面(highly adaptive contextual interfaces)能根据不同设备的屏幕大小、分辨率和使用情境动态调整界面布局和显示内容，使用户在小屏幕设备(如手机、手表)上也能获得良好的使用体验，因此能有效应对设备硬件限制.
(T) 6. 使用高度自适应的上下文界面是应对设备限制(如屏幕尺寸)的一个好方法.

( ) 18-7. Which of the following are common MobileApp design mistakes. 238

A. Inconsistency
B. Interoperability
C. Lean design
D. Overdesigning

A选项“Inconsistency”(不一致性)指界面、操作或风格不统一，容易让用户困惑．D选项“Overdesigning”(过度设计)是功能或界面过于复杂，超出用户需求，会影响用户体验．这两项是常见的设计错误．B选项“Interoperability”(互操作性)和C选项“Lean design”(精益设计)是好的设计目标，不属于设计错误.
(AD) 7. 以下哪些是常见的移动应用设计错误.
A. **不一致性**
B. 互操作性
C. 精益设计
D. **过度设计**

( ) 18-8. It is better to multiple short pages than long scrolling forms when implementing mobile device user interfaces. 239

该题考查移动设备用户界面设计原则．将表单分成多个短页面会增加用户的点击和跳转，降低效率和连贯性．适当使用长滚动页面能让用户更顺畅地填写信息，减少页面切换，更符合移动设备的交互习惯．因此，多个短页面不如长滚动表单更好.
(F) 8. 在实现移动设备用户界面时，使用多个简短页面比使用长滚动表单更好.

( ) 18-9. Java is the best programming language to use when you want to create portable MobileApps. 240

虽然Java常用于移动应用开发，尤其是Android，但它不是开发可移植(portable)移动应用的最佳选择．可移植性指应用能在不同平台(如iOS和Android)运行．为了实现更好的可移植性，通常会选用React Native、Flutter等跨平台开发框架，而不是仅使用针对某一平台(如Android)的Java．因此，Java不是开发可移植移动应用的最佳语言.
(F) 9. 当你想要创建可移植的移动应用程序时，Java是最好的编程语言.

( ) 18-10. Service computing allows you to avoid the need to integrate service source code into the mobile device client. 241

服务计算(Service computing)允许你不需要将服务的源代码集成到移动设备客户端中．服务通常以远程接口形式提供，客户端通过网络调用这些服务，无需获取或集成其源代码，这样能降低客户端复杂性，提高灵活性和可维护性.
(T) 10. 服务计算使您无需将服务源代码集成到移动设备客户端中.

( ) 18-11. The most important MobileApp architecture decision whether to build a thin or fat mobile client. 242

移动应用架构设计中，决定构建瘦客户端(Thin Client)还是胖客户端(Fat Client)非常重要．瘦客户端把业务逻辑和数据处理放在服务器端，客户端只负责展示和交互；胖客户端则把处理和存储更多放在本地设备．这个选择会直接影响应用的性能、扩展性、离线能力、更新难度和用户体验，是移动应用架构中的关键决策.
(T) 11. 移动应用程序架构中最重要的决策是选择构建瘦客户端还是胖客户端.

( ) 19-1. Quality of conformance focuses on the degree to which the implementation of a design meets its requirements and performance goals. 243

Quality of conformance(符合性质量)指实际开发的软件产品在实现过程中有多大程度上符合最初的设计要求和性能目标．实现与设计要求和预定目标一致，说明质量符合性高.
(T) 1. 一致性质量关注设计实现满足其需求和性能目标的程度.

( ) 19-2. Which of the following is not one of the attributes of software quality? 244

A. Adds value for developers and users
B. Effective software process creates infrastructure
C. Removes need to consider performance issues
D. Useful products satisfy stakeholder requirements

A、B、D选项描述了软件质量的相关属性，比如为用户和开发者增加价值、有效的软件过程、满足利益相关者需求等. C选项"消除了考虑性能问题的需要"不正确，因为性能是软件质量的重要属性之一，不能忽略. 高质量软件必须考虑性能问题，而不是不需要考虑. 因此，C不是软件质量的属性.

(C) 2. 下列哪一项不是软件质量的属性?
A. 为开发者和用户增加价值
B. 高效的软件过程创建基础设施
C. **消除对性能问题的考虑需求**
D. 有用的产品满足相关方需求

( ) 19-3. Product quality can only be assessed by measuring hard quality factors. 245

产品质量评估不仅依赖硬性质量因素(如性能、可靠性等可量化指标)，还包括软性质量因素(如用户满意度、易用性、可维护性等主观或难以量化的指标). 因此，不能只依靠硬性质量因素来评估产品质量.

(T) 3. 只能通过测量硬性质量因素来评估产品质量.

( ) 19-4. Many software metrics can only be measured indirectly. 246

很多软件度量只能间接测量. 许多软件属性，如可维护性、复杂度、可靠性等，无法直接用工具或方法测量，通常需要通过其他可量化的指标(如代码行数、缺陷数、模块之间的耦合度等)间接反映这些属性. 因此，很多软件度量确实只能间接测量.

(T) 4. 许多软件度量只能间接测量.

( ) 19-5. Which of the following are ISO 9126 software quality factors? 247

A. Functionality
B. Portability
C. Reliability
D. Visual appeal

ISO 9126软件质量模型将软件质量分为六大特性：功能性(Functionality)、可靠性(Reliability)、易用性(Usability)、效率(Efficiency)、可维护性(Maintainability)和可移植性(Portability). 选项A(功能性)、B(可移植性)、C(可靠性)属于ISO 9126定义的质量特性，D(视觉吸引力)不属于ISO 9126中的质量特性.

(ABC) 5. 以下哪些是ISO 9126软件质量特性?
A. **功能性**
B. **可移植性**
C. **可靠性**
D. 视觉吸引力

( ) 19-6. Developers need to create a collection of targeted questions to asses each quality factor. 248

开发人员需要为每个质量因素设计有针对性的问题进行评估. 软件质量包括可靠性、可维护性、安全性等多个方面. 针对每个因素制定具体问题，有助于有效检查和改进相应质量，保证软件整体质量.

(T) 6. 开发人员需要创建一组有针对性的问题来评估每一个质量因素.

( ) 19-7. Software metrics represent direct measures of some manifestation of quality. 249

软件度量(software metrics)通常是对软件过程、产品或项目特征的间接度量，而非直接度量质量本身. 例如代码行数、缺陷数、复杂度等指标，并不能直接反映软件质量，只是借助这些量化数据间接推断质量状况. 因此，软件度量是质量的间接度量，不是直接度量.

(F) 7. 软件度量代表了某些质量表现的直接度量.

( ) 19-8. The quality dilemma might be summarized as choosing between building things quickly or building things correctly. 250

质量困境(quality dilemma)并不是只能选择"快"或"对". 软件工程强调在保证质量的同时提高开发效率，优秀的软件开发流程和方法能兼顾开发速度与软件质量，两者不应对立.

(F) 8. 质量困境可以概括为在快速构建和正确构建之间进行选择.

( ) 19-9. Good enough software delivers high quality software functions along with specialized functions that contain known bugs. 251

"Good enough software"既提供高质量的核心功能，也允许某些特定功能存在已知bug. 在实际开发中，为了满足进度和需求，通常优先保证主要功能的高质量和稳定性，对于不太重要或很少使用的特殊功能，即使有已知缺陷，也会暂时保留. 这种做法是为了在有限的时间和资源下，先保证软件可用，后续再逐步完善.

(T) 9. 足够好的软件在交付高质量软件功能的同时，也包含已知缺陷的专用功能.

( ) 19-10. Which of the following is likely to be the most expensive cost of quality? 252

A. Appraisal costs
B. External failure costs
C. Internal failure costs
D. Prevention costs

质量成本包括预防成本、鉴定成本、内部故障成本和外部故障成本. 外部故障成本是指产品交付客户后出现问题产生的费用，如召回、赔偿、声誉损失等. 这些问题影响客户和市场，修复和补救的代价最大，因此外部故障成本通常是最高的.

(B) 10. 下列哪项最有可能是质量成本中最昂贵的?
A. 评估成本
B. **外部故障成本**
C. 内部故障成本
D. 预防成本

( ) 19-11. Poor quality leads to software risks that can become serious? 253

软件质量差会导致系统崩溃、数据丢失、安全漏洞等严重风险. 如果忽视质量，问题发生的概率和影响都会增加，可能导致项目失败或损失. 因此，确保高质量是降低软件风险的关键.

(T) 11. 低质量会导致软件风险，这些风险可能变得严重吗?

( ) 19-12. When a system fails to deliver required functions it is because the customer changes requirements? 254

系统未能实现所需功能的原因有很多，不仅仅是客户变更需求. 还可能是需求分析不充分、设计或开发出错、测试不到位、沟通不畅等. 客户变更需求只是众多可能原因之一.

(F) 12. 当系统未能交付所需功能时，是因为客户更改了需求吗?

( ) 19-13. Developers must start focusing on quality during the design phase in order to build secure systems. 255

开发人员需要从设计阶段关注质量，才能构建安全系统. 软件的质量和安全性很大程度上决定于设计阶段的决策. 如果在设计阶段就考虑安全性和质量，如合理架构、模块划分、数据验证等，可以有效预防后续开发中的漏洞和缺陷. 如果等到开发后期或测试阶段再考虑质量和安全，会导致修复成本高且难以彻底解决问题. 因此，从设计阶段关注质量是实现安全系统的关键.

(T) 13. 开发人员必须在设计阶段开始关注质量，以构建安全的系统.

( ) 19-14. Which of the following management decisions have the potential to impact software quality? 256

A. Estimation decisions
B. Risk-oriented decisions
C. Scheduling decisions
D. All of the above

估算决策(如项目成本和资源的估算)、风险导向决策(如应对潜在项目风险的方法)、进度安排决策(如项目时间表制定)都会直接或间接影响软件质量. 估算不准确、风险管理不到位或进度安排不合理都可能导致软件质量下降，因此所有这些管理决策都可能影响软件质量.

(D) 14. 以下哪些管理决策有可能影响软件质量?
A. 估算决策
B. 以风险为导向的决策
C. 进度安排决策
D. **以上所有**

( ) 19-15. The project plan should include explicit techniques for _____ and _____ management? 257

A. change
B. cost
C. error
D. quality

项目计划中应明确变更管理(change management)和质量管理(quality management)技术. 项目实施过程中变化不可避免，需要变更管理来应对；同时，软件质量影响产品可用性和用户满意度，需要质量管理来保证. 这两项是项目计划中必须明确管理的内容.

(AD) 15. 项目计划应包括明确的_____和_____管理技术?
A. **变更**
B. 成本
C. 错误
D. **质量**

( ) 19-16. Quality control encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals. 258

质量控制(Quality control)是在软件工程过程中，通过评审、测试、检查等活动，确保每个工作成果(work product)达到预定质量目标. 它关注于发现和纠正产品缺陷，从而保证最终交付的软件质量. 题目描述准确.

(T) 16. 质量控制包括一组软件工程活动，这些活动有助于确保每个工作产品都能达到其质量目标.

( ) 19-17. The goal of quality assurance to insure that a software project is error free. 259

质量保证(quality assurance, QA)的目标是确保软件开发过程和产品符合既定的质量标准和要求，尽量减少缺陷，但无法保证软件绝对没有错误. 软件很难做到完全无错，QA侧重于预防和发现问题，而不是保证百分之百无错.

(F) 17. 质量保证的目标是确保一个软件项目没有错误.

( ) 20-1. The purpose of software reviews is to uncover errors and defects in work products so they can be removed before moving on to the next phase of development. 260

软件评审(software reviews)的主要目的是在开发流程各阶段及时发现和修正工作产品(如需求文档、设计文档、代码等)中的错误和缺陷. 这样可以在进入下一个开发阶段前，减少后续修复缺陷的成本和风险，提高软件质量和开发效率.

(T) 1. 软件评审的目的是发现工作产品中的错误和缺陷，以便在进入开发的下一个阶段之前将其消除.

( ) 20-2. In general the earlier a software defect is discovered and corrected the less costly to the overall project budget. 261

软件开发中，越早发现和修复缺陷，所需的人力、物力和时间成本越低. 如果等到后期甚至上线后才发现问题，修复时不仅要修改代码，还可能需要重新设计、重新测试，甚至修复已经造成的损失，成本会大大增加. 早发现、早修复缺陷有助于节约项目整体成本.

(T) 2. 一般来说，软件缺陷被发现和纠正得越早，对整个项目预算的成本就越低.

( ) 20-3. Defect amplification models can be used to illustrate the costs associated with using software from its initial deployment to its retirement. 262

缺陷放大模型(defect amplification models)主要说明在软件开发过程中，缺陷如果没有及时发现和修复，会在后续阶段被放大，修复成本也会增加．它关注开发过程中的缺陷流转和放大，不涉及整个软件生命周期(从初始部署到退役)中使用软件的成本，因此不适用于说明软件从部署到退役期间的使用成本．

(F) 3. 缺陷放大模型可以用来说明从软件初始部署到退役期间使用软件所产生的成本．

(　) 20-4. Review metrics can be used to assess the efficacy of each review activity. 263

评审指标(review metrics)可以用于评估每次评审活动的有效性．在软件工程中，通过收集和分析评审过程中的数据，如发现的缺陷数量、评审所用时间等，可以判断评审活动是否达到预期效果，并为持续改进评审流程提供依据．

(T) 4. 评审度量可以用来评估每个评审活动的有效性．

(　) 20-5. Defect density can be estimated for any software engineering work product. 264

缺陷密度(Defect Density)指的是在软件工程的任何工作产品(如需求文档、设计文档、代码、测试用例等)中，每单位大小(如每千行代码或每页文档)发现的缺陷数量．所有工作产品都可能有缺陷，因此理论上都能估算其缺陷密度，这有助于衡量和改进软件质量．

(T) 5. 缺陷密度可以用于估算任何软件工程工作产品．

(　) 20-6. Agile software developers are aware that software reviews always take time without saving any. 265

敏捷开发者认为软件评审虽然会花时间，但能提前发现和修正问题，从而在后续开发和维护阶段节省更多时间和成本，因此软件评审是有价值的．

(F) 6. 敏捷软件开发人员意识到，软件评审总是会花费时间，而不会节省任何时间．

(　) 20-7. The level of review formality is determined by which of the following? 266

A. amount of preparation
B. reviewer follow-up
C. size of project budget
D. structure of review

评审的正式程度与以下因素有关：A. amount of preparation(准备工作的多少)，准备越充分，评审通常越正式；B. reviewer follow-up(评审人员的后续跟进)，有无后续跟进是正式评审的重要标志；D. structure of review(评审的结构)，评审流程越有结构性，正式程度越高．C选项size of project budget(项目预算大小)可能影响项目管理的复杂度，但与评审正式程度没有直接关系．

(ABD) 7. 审查的正式程度由下列哪些因素决定?
A. **准备工作的量**
B. **审查者的后续跟进**
C. 项目预算的规模
D. **审查的结构**

(　) 20-8. An informal review may consist of which of the following? 267

A. casual meeting
B. desk check
C. inspection
D. pair programming

非正式评审(informal review)是指不需要严格流程和文档记录的评审，主要目的是快速发现和解决问题．A"casual meeting"是随意的会议，B"desk check"是同事间互查代码或文档，这两种方式属于非正式评审．C"inspection"有正式流程和记录，属于正式评审．D"pair programming"是开发方式，不算评审．

(AB) 8. 非正式评审可能包括以下哪项?
A. **非正式会议**
B. **桌面检查**
C. 评审
D. 结对编程

(　) 20-9. Which of the following are objectives for formal technical reviews? 268

A. allow senior staff members to correct errors
B. assess programmer productivity
C. determining who introduced an error into a program
D. uncover errors in software work products

正式技术评审(formal technical reviews, FTR)的主要目的是在软件开发早期发现和纠正软件工作产品中的错误，提高软件质量．选项A(让高级员工纠错)、B(评估程序员效率)、C(找出错误责任人)都不是FTR的目标．FTR强调团队合作和产品质量，而不是个人表现或责任追究．

(D) 9. 下列哪些是正式技术评审的目标?
A. 允许高级员工纠正错误
B. 评估程序员的生产力
C. 确定是谁在程序中引入了错误
D. **发现软件工作产品中的错误**

(　) 20-10. At the end of a formal technical review all attendees can decide to 269

A. accept the work product without modification
B. modify the work product and continue the review
C. reject the product due to stylistic discrepancies
D. reject the product due to severe errors

正式技术评审(Formal Technical Review, FTR)结束时，参与者可以选择接受工作产品且无需修改(A)，这说明没有发现严重问题；也可以因存在严重错误而拒绝产品(D)，说明产品有重大缺陷需返工．继续评审(B)通常不符合FTR流程，评审结束时应有明确结论．仅因格式或风格问题拒绝产品(C)不合理，这些问题通常可通过小修改解决，不影响产品实质内容．

(AD) 10. 在一次正式的技术评审结束时，所有与会者可以决定
A. **无修改地接受工作产品**
B. 修改工作产品并继续评审
C. 因风格差异而拒绝该产品
D. **因严重错误而拒绝该产品**

(　) 20-11. A review summary report answers which three questions? 270

A. terminate project, replace producer, request a time extension
B. what defects were found, what caused defects, who was responsible
C. what was reviewed, who reviewed it, what were the findings
D. none of the above

评审总结报告(review summary report)主要记录和汇总评审过程中的关键信息，包括：评审了什么内容(what was reviewed)、参与评审的人员(who reviewed it)、以及评审中发现的问题或结论(what were the findings)．其他选项涉及责任归属或项目决策，与评审总结报告的主要用途无关．

(C) 11. 评审总结报告回答哪三个问题?
A. 终止项目、替换生产者、请求延期
B. 发现了哪些缺陷、缺陷的原因是什么、谁负责
C. **评审了什么、谁进行了评审、评审结果是什么**
D. 以上都不是

(　) 20-12. In any type of technical review, the focus of the review is on the product and not the producer. 271

在技术评审(technical review)中，关注点是被评审的产品(如需求文档、设计、代码等)，而不是开发者个人．这样可以客观发现并改进产品中的问题，避免对个人的批评，有助于营造良好团队氛围，提高软件质量．

(T) 12. 在任何类型的技术评审中，评审的重点是产品本身，而不是产品的生产者．

(　) 20-13. Sample driven reviews only make sense for very small software development projects. 272

样本驱动评审(Sample driven reviews)是一种通过抽查部分工作产品来发现缺陷的方法，不只适用于小型项目，也适用于中大型项目．对大型项目来说，全面检查成本高，样本评审可以在较短时间内发现共性问题，提高评审效率，因此大项目同样适用．

(F) 13. 仅对非常小型的软件开发项目，样本驱动评审才有意义．

(　) 21-1. Software quality might be defined as conformance to explicitly stated requirements and standards, nothing more and nothing less. 273

软件质量不仅指符合明确规定的需求和标准，还包括可维护性、易用性、性能、安全性等其他方面．有些重要的质量属性可能没有在需求文档或标准中体现，但也非常关键．因此，只看"符合需求和标准"无法全面反映软件的实际质量．

(F) 1. 软件质量可以被定义为符合明确规定的需求和标准，仅此而已．

(　) 21-2. People who perform software quality assurance must look at the software from the customer's perspective. 274

软件质量保证(Software Quality Assurance，SQA)人员在工作时要站在客户的角度来看待软件．SQA的核心目标是确保软件能满足客户需求和期望，保证软件质量．如果只从开发者的角度考虑，可能会忽略客户实际使用中的问题．因此，SQA人员需要模拟客户的使用场景，发现潜在缺陷，确保软件的功能、性能和用户体验符合客户要求．

(T) 2. 执行软件质量保证的人必须从客户的角度来看待软件．

(　) 21-3. The elements of software quality assurance consist of reviews, audits, and testing. 275

软件质量保证(Software Quality Assurance，SQA)不仅包括评审(reviews)、审计(audits)和测试(testing)，还涉及标准制定、过程改进、配置管理、文档控制、培训等多种活动．题目中的说法不全面，因此是错误的．

(F) 3. 软件质量保证的要素包括评审、审计和测试．

(　) 21-4. Which of these activities is not one of the activities recommended to be performed by an independent SQA group? 276

A. prepare SQA plan for the project
B. review software engineering activities to verify process compliance
C. report any evidence of noncompliance to senior management
D. serve as the sole test team for any software produced

D项"作为任何开发软件的唯一测试团队"不是独立SQA(软件质量保证，Software Quality Assurance)小组推荐的活动．SQA小组的职责包括制定SQA计划、监督和审核软件工程活动过程是否合规，以及向高层管理层报告不合规情况，而不是直接负责所有测试工作．测试通常由专门的测试团队或开发团队执行，SQA主要负责监督和保证质量．

(D) 4. 以下哪项活动不是独立SQA小组建议执行的活动?
A. 为项目制定SQA计划
B. 审查软件工程活动以验证过程合规性
C. 向高级管理层报告任何不合规的证据
D. **作为所开发软件的唯一测试团队**

(　) 21-5. Metrics can be used to indicate the relative strength of a software quality attribute. 277

度量指标(metrics)用于表示软件质量属性的相对强弱．软件工程中常用如缺陷密度、代码复杂度、响应时间等度量，量化和比较软件的可维护性、可靠性、效率等质量属性．这些指标能客观反映和比较某一质量属性的高低或优劣，因此度量可以指示质量属性的相对强度．

(T) 5. 度量可以用来指示软件质量属性的相对强度．

( ) 21-6. Attempts to apply mathematical proof to demonstrate that a program conforms to its specifications are doomed to failure. 278

实际上，应用数学证明(如形式化方法)在某些情况下可以证明程序符合其规格，特别是在安全关键或对正确性要求很高的系统中．虽然这些方法实现有难度和局限性，但并非必然失败．
(F) 6. 试图应用数学证明来证明程序符合其规格的做法注定会失败．

( ) 21-7. Statistical quality assurance involves 279

A. using sampling in place of exhaustive testing of software
B. surveying customers to find out their opinions about product quality
C. tracing each defect to its underlying cause, isolating the "vital few" causes, and moving to correct them
D. tracing each defect to its underlying causes and using the Pareto principle to correct each problem found

统计质量保证(statistical quality assurance, SQA)主要是通过分析缺陷，追踪每个缺陷的根本原因，找出真正影响质量的"关键少数"原因，并有针对性地改进．A描述的是抽样测试，B是用户调研，D对Pareto原则的理解不准确．
(C) 7. 统计质量保证包括
A. 使用抽样代替对软件的穷举测试
B. 调查客户以了解他们对产品质量的看法
C. **追踪每个缺陷的根本原因，找出"关键少数"原因，并采取措施加以纠正**
D. 追踪每个缺陷的根本原因，并利用帕累托原则纠正发现的每个问题

( ) 21-8. Six Sigma methodology defines three core steps. 280

A. analyze, improve, control
B. analyze, design, verify
C. define, measure, analyze
D. define, measure, control

Six Sigma方法论的三个核心步骤是define(定义)、measure(测量)、analyze(分析)，简称DMA(Define, Measure, Analyze)．这三个步骤包括明确问题、收集和测量相关数据、分析数据找出问题根源．
(C) 8. 六西格玛方法论定义了三个核心步骤．
A. 分析、改进、控制
B. 分析、设计、验证
C. **定义、测量、分析**
D. 定义、测量、控制

( ) 21-9. Software reliability problems can almost always be traced to 281

A. errors in accuracy
B. errors in design
C. errors in implementation
D. errors in operation

软件可靠性问题通常源于设计错误和实现错误．设计错误指系统结构、流程或算法设计阶段出现的问题，会影响系统的正确性．实现错误指编码或具体实现过程中产生的bug或失误，这两类错误直接影响软件能否按预期工作，是导致软件不可靠的主要原因．A和D主要涉及具体数据或用户操作，不是可靠性问题的根本原因．
(BC) 9. 软件可靠性问题几乎总是可以追溯到
A. 精度错误
B. **设计错误**
C. **实现错误**
D. 操作错误

( ) 21-10. Software safety is a quality assurance activity that focuses on hazards that 282

A. affect the reliability of a software component
B. may cause an entire system to fail
C. may result from user input errors
D. prevent profitable marketing of the final product

软件安全(software safety)在质量保证中主要关注可能导致整个系统失效的潜在危险(hazards)，如系统崩溃或产生严重后果的错误．B选项"may cause an entire system to fail"正是软件安全关注的核心问题．其他选项提到的可靠性、用户输入错误或市场营销问题，不属于软件安全的主要关注点．
(B) 10. 软件安全是一项质量保证活动，其重点关注那些可能导致的危害
A. 影响软件组件可靠性的危害
B. **可能导致整个系统失效的危害**
C. 可能由用户输入错误引起的危害
D. 阻碍最终产品盈利性营销的危害

( ) 21-11. The ISO quality assurance standard that applies to software engineering is 283

A. ISO 9000
B. ISO 9001
C. ISO 9002
D. ISO 9003

ISO 9001是ISO 9000系列中专门针对质量管理体系要求的标准，适用于包括软件开发在内的各行业．它强调过程管理和持续改进，是软件企业建立和改进质量管理体系的主要依据．其他选项如ISO 9000是总纲，ISO 9002和9003是已被淘汰的具体标准，不直接针对软件过程．
(B) 11. 适用于软件工程的ISO质量保证标准是
A. ISO 9000
B. **ISO 9001**
C. ISO 9002
D. ISO 9003

( ) 21-12. Which of the following is not a section in the standard for SQA plans recommended by IEEE? 284

A. budget
B. documentation
C. reviews and audits
D. test

IEEE推荐的软件质量保证(SQA)计划标准包括文档(documentation)、评审与审计(reviews and audits)、测试(test)等内容，不包括预算(budget)．预算属于项目管理，不属于SQA计划的标准内容．
(A) 12. 下列哪一项不是IEEE推荐的软件质量保证(SQA)计划标准中的部分?
A. **预算**
B. 文档
C. 评审与审计
D. 测试

( ) 22-1. In software quality assurance work there is no difference between software verification and software validation. 285

软件验证(verification)和软件确认(validation)有区别．验证关注软件是否被正确地构建，即是否符合规格说明；确认关注软件是否构建了正确的内容，即是否满足用户需求．验证强调做对，确认强调做对的事．两者在软件质量保证工作中侧重点不同．
(F) 1. 在软件质量保证工作中，软件验证和软件确认没有区别．

( ) 22-2. The best reason for using Independent software test teams is that 286

A. software developers do not need to do any testing
B. strangers will test the software mercilessly
C. testers do not get involved with the project until testing begins
D. the conflicts of interest between developers and testers is reduced

独立的软件测试团队可以减少开发人员和测试人员之间的利益冲突．开发人员编写代码后，可能对自己的代码存在偏见，容易忽略或下意识避开一些缺陷．独立测试团队不参与开发，能以更加客观和全面的角度发现问题，提高测试的公正性和有效性．
(D) 2. 使用独立软件测试团队的最佳理由是
A. 软件开发人员不需要进行任何测试
B. 外部人员会无情地测试软件
C. 测试人员直到测试开始才参与项目
D. **减少了开发人员和测试人员之间的利益冲突**

( ) 22-3. What is the normal order of activities in which traditional software testing is organized? 287

A. integration testing, system testing, unit testing, validation testing.
B. unit testing, validation testing, system testing, integration testing
C. unit testing, integration testing, validation testing, system testing
D. validation testing, system testing, integration testing, unit testing

传统软件测试的顺序是：先进行单元测试(unit testing)，测试最小的代码单元；再做集成测试(integration testing)，测试单元组合后的功能；然后进行确认测试(validation testing)，检查软件是否满足需求；最后进行系统测试(system testing)，对整个系统进行全面测试．
(C) 3. 传统软件测试通常按照什么顺序组织各项活动?
A. 集成测试，系统测试，单元测试，确认测试．
B. 单元测试，确认测试，系统测试，集成测试．
C. **单元测试，集成测试，确认测试，系统测试．**
D. 确认测试，系统测试，集成测试，单元测试．

( ) 22-4. By collecting software metrics and making use of existing software reliability models it is possible to develop meaningful guidelines for determining when software testing is done. 288

通过收集软件度量(如缺陷密度、测试覆盖率等)并结合已有的软件可靠性模型，可以量化软件质量和可靠性，从而制定更科学的标准判断测试是否结束．这些数据和模型为判断测试充分与否、软件能否交付提供客观依据，因此能够制定有意义的测试完成标准．
(T) 4. 通过收集软件度量并利用现有的软件可靠性模型，可以制定有意义的指导方针来确定软件测试何时完成．

( ) 22-5. Which of the following strategic issues needs to be addressed in a successful software testing process? 289

A. conduct formal technical reviews prior to testing
B. specify requirements in a quantifiable manner
C. use independent test teams
D. wait till code is written prior to writing the test plan

A(测试前进行正式技术评审)和B(以可量化方式规定需求)是重要的战略措施．A有助于早期发现缺陷，降低后期修复成本；B确保测试目标明确且可验证，提高测试有效性．C(使用独立测试团队)虽然能提升测试客观性，但属于组织层面的选择，并非所有项目的必要战略条件．D(等代码写完才写测试计划)是错误做法，测试计划应尽早制定，贯穿开发流程，以有效发现和预防缺陷．因此，A和B属于优先解决的战略性问题．
(AB) 5. 在成功的软件测试过程中，以下哪些战略性问题需要被解决?
A. **在测试之前进行正式的技术评审**
B. **以可量化的方式规定需求**
C. 使用独立的测试团队
D. 等到代码编写完成后再编写测试计划

( ) **22-6. Which of the following need to be assessed during unit testing?** 290

A. algorithmic performance
B. code stability
C. error handling
D. execution paths

单元测试主要关注代码的正确性和健壮性. C(错误处理)需要评估, 因为要确保模块能正确处理各种异常和错误情况. D(执行路径)也需评估, 因为要验证所有代码路径都能按预期工作. A(算法性能)通常在性能测试阶段关注, B(代码稳定性)属于集成或系统测试时关注的内容, 不是单元测试的主要目标.

(CD) 6. 在单元测试期间需要评估以下哪些内容?
A. 算法性能
B. 代码稳定性
C. **错误处理**
D. **执行路径**

( ) **22-7. Units and stubs are not needed for unit testing because the modules are tested independently of one another.** 291

在单元测试中, 模块虽然独立测试, 但常常依赖其他模块的输入或输出. 为了隔离被测试模块, 需要用桩(stubs)和驱动(drivers/units)来模拟尚未开发或不可用模块的行为, 从而单独测试当前模块. 因此, 单元测试通常需要桩和驱动.

(F) 7. 单元测试不需要单元和桩, 因为各个模块是相互独立测试的.

( ) **22-8. Top-down integration testing has as it's major advantage(s) that** 292

A. low level modules never need testing
B. major decision points are tested early
C. no drivers need to be written
D. no stubs need to be written

自顶向下集成测试(Top-down integration testing)从高层模块开始, 逐步集成和测试底层模块. 其优点包括: B. 主要决策点(major decision points)能在早期测试, 因为高层模块通常包含系统的主要控制逻辑. C. 不需要编写驱动程序(drivers), 因为主控模块本身可以调用下层模块进行测试. A错误, 低层模块依然需要测试, 只是顺序不同. D错误, 因为底层模块未实现时, 仍需用桩模块(stubs)模拟其行为.

(BC) 8. 自顶向下集成测试的主要优点是
A. 低层模块无需测试
B. **主要决策点可以及早测试**
C. **无需编写驱动程序**
D. 无需编写桩程序

( ) **22-9. Bottom-up integration testing has as it's major advantage(s) that** 293

A. major decision points are tested early
B. no drivers need to be written
C. no stubs need to be written
D. regression testing is not required

自底向上的集成测试从最底层模块开始, 逐步集成到上层模块. 每次集成上层模块时, 底层模块已经存在并可以运行, 所以不需要为下层模块编写桩(stub), 这就是选项C的含义. 而驱动程序(drivers)仍然需要, 因为高层模块还未集成时, 需要驱动来调用底层模块.

(C) 9. 自底向上的集成测试的主要优点是
A. 主要决策点可以早期测试
B. 不需要编写驱动程序
C. **不需要编写桩程序**
D. 不需要回归测试

( ) **22-10. Regression testing should be a normal part of integration testing because as a new module is added to the system new** 294

A. control logic is invoked
B. data flow paths are established
C. drivers require testing
D. all of the above

集成测试时, 每加入新模块, 系统的控制逻辑和数据流路径都会变化, 因此需要回归测试, 确保新模块没有破坏已有功能. A(新的控制逻辑被调用)和B(新的数据流路径被建立)体现了这种变化. C选项"驱动程序需要测试"并不是每次集成都必须的, 所以不选C.

(AB) 10. 回归测试应当作为集成测试的常规部分, 因为当新模块被添加到系统中时, 会有新的
A. **控制逻辑被调用**
B. **数据流路径被建立**
C. 驱动程序需要测试
D. 以上所有

( ) **22-11. Smoke testing might best be described as** 295

A. bulletproofing shrink-wrapped software
B. rolling integration testing
C. testing that hides implementation errors
D. unit testing for small programs

Smoke testing是一种初步集成测试, 用于软件新版本集成后快速验证核心功能是否正常, 确保软件基本可用再进行深入测试. 选项B"rolling integration testing"符合烟雾测试的特点, 因为它强调在集成过程中持续快速测试, 及时发现严重错误. 其他选项不准确描述smoke testing.

(B) 11. 冒烟测试最好的描述是
A. 对封装软件进行防弹测试
B. **持续集成测试**
C. 隐藏实现错误的测试
D. 对小型程序进行单元测试

( ) **22-12. When testing object-oriented software it is important to test each class operation separately as part of the unit testing process.** 296

在面向对象软件测试中, 单元测试的基本单位通常是"类"而不是单独的"类操作"(即类中的每个方法). 应测试类的所有操作及其交互, 而不仅仅是单独测试每个操作, 因为对象的状态和方法之间通常紧密关联. 只测试每个操作, 可能无法覆盖对象状态变化和方法之间的协作问题.

(F) 12. 在测试面向对象的软件时, 重要的是在单元测试过程中分别测试每个类的操作.

( ) **22-13. The OO testing integration strategy involves testing** 297

A. groups of classes that collaborate or communicate in some way
B. single operations as they are added to the evolving class implementation
C. operator programs derived from use-case scenarios
D. none of the above

面向对象(OO)测试中的集成测试策略侧重于类之间的协作关系. 在软件系统中, 类通常与其他类交互, 而不是单独工作. 选项A描述了测试协作或通信的类组, 这可以发现它们之间接口和消息传递等问题. 其他选项关注单个操作、用例相关程序或不相关内容, 因此不正确.

(A) 13. 面向对象测试的集成策略涉及测试
A. **以某种方式协作或通信的类的组**
B. 在不断演化的类实现中添加的单个操作
C. 从用例场景中派生的操作程序
D. 以上都不是

( ) **22-14. Since many WebApps evolve continuously, the testing process must be ongoing as well.** 298

很多Web应用(WebApps)会持续演化和更新, 所以测试过程也必须持续进行. 每当Web应用有新功能、修改或修复上线时, 可能会引入新缺陷或影响已有功能, 因此需要不断测试, 确保软件质量和稳定性. 这也是敏捷开发和持续集成中的重要做法.

(T) 14. 由于许多Web应用程序不断演化, 测试过程也必须是持续进行的.

( ) **22-15. Testing MobileApps is not different than testing WebApps.** 299

测试移动应用和Web应用存在很大不同. 移动应用需要考虑不同操作系统(如iOS和Android)、各种屏幕尺寸和硬件性能等, 而Web应用主要运行在浏览器上, 更关注浏览器兼容性. 二者在运行环境、用户交互、设备多样性和网络状况等方面也不同, 因此测试方法和重点不一样.

(F) 15. 测试移动应用程序与测试Web应用程序没有区别.

( ) **22-16. The focus of validation testing is to uncover places that s user will be able to observe failure of the software to conform to its requirements.** 300

验证测试(validation testing)主要目的是确认软件是否满足用户需求, 即检验软件是否"做了用户想要的事". 因此, 验证测试关注发现用户能够察觉到软件未能实现需求的地方. 题目描述和验证测试的定义一致.

(T) 16. 验证测试的重点是发现用户能够观察到软件未能符合其需求的地方.

( ) **22-17. Software validation is achieved through a series of tests performed by the user once the software is deployed in his or her work environment.** 301

题干认为软件验证是指用户在软件部署到实际工作环境后, 通过一系列测试来完成, 这是错误的. 软件验证(Software validation)是在软件开发阶段, 通过系统测试、验收测试等, 确保软件满足用户需求, 通常在正式部署前完成. 用户在部署后进行的测试一般是操作使用或维护阶段的内容, 不属于软件验证.

(F) 17. 软件验证是通过用户在其工作环境中部署软件后进行的一系列测试来实现的.

( ) **22-18. Configuration reviews are not needed if regression testing has been rigorously applied during software integration.** 302

配置评审和回归测试是不同的活动. 回归测试关注修改后软件是否有新错误, 配置评审则系统检查软件配置项(如代码、文档、版本)的一致性、完整性和正确性, 确保配置管理规范和产品质量. 即使严格回归测试, 也不能替代配置评审.

(F) 18. 如果在软件集成过程中已经严格进行了回归测试, 则不需要进行配置评审.

( ) **22-19. Acceptance tests are normally conducted by the** 303

A. developer
B. end users
C. test team
D. systems engineers

验收测试(Acceptance tests)通常由最终用户(end users)进行, 目的是确认软件是否满足用户需求和业务目标, 只有最终用户最清楚需求是否被实现. 开发者和测试团队只是辅助, 主要执行工作由用户或用户代表完成.

(B) 19. 验收测试通常由谁来进行
A. 开发人员
B. **最终用户**
C. 测试团队
D. 系统工程师

( ) **22-20. Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that software is able to continue execution without interruption.** 304

恢复测试(Recovery testing)关注软件在发生故障后能否正确恢复到正常工作状态, 不是要求错误发生时不中断执行. 题目描述有误.

(F) 20. 恢复测试是一种系统测试, 它通过多种方式强制软件发生故障, 并验证软件是否能够不中断地继续执行.

( ) 22-21. Security testing attempts to verify that protection mechanisms built into a system protect it from improper penetration. 305

安全测试(Security testing)的主要目标是验证系统内置的保护机制能否防止不当入侵. 它检查系统是否能抵抗未授权访问、数据泄露等威胁, 以确保系统安全性.

(T) 21. 安全性测试旨在验证系统中内置的保护机制能否防止不当入侵.

( ) 22-22. Stress testing examines the pressures placed on the user during system use in extreme environments. 306

压力测试(Stress Testing)是在极端条件下测试软件系统, 以确认其性能、稳定性和可靠性. 它关注系统在超负荷或极端资源消耗下的表现, 不涉及用户在极端环境下的心理或生理压力. 题目将压力测试理解为考察用户压力是错误的, 正确的做法是考察系统本身的承受能力.

(F) 22. 压力测试检查在极端环境下系统使用时对用户施加的压力.

( ) 22-23. Performance testing is only important for real-time or embedded systems. 307

性能测试不仅对实时系统或嵌入式系统重要, 任何类型的软件, 如网站、企业应用、移动应用等, 都需要性能测试, 以确保系统在高负载下正常运行, 响应速度满足用户需求. 性能问题会影响用户体验和系统可靠性, 因此各类软件都需要进行性能测试.

(F) 23. 性能测试只对实时或嵌入式系统重要.

( ) 22-24. Debugging is not testing, but always occurs as a consequence of testing. 308

调试(Debugging)和测试(Testing)是软件工程中不同的活动. 测试是运行程序并发现错误或缺陷, 调试是在发现错误后, 分析原因并修正这些错误. 调试通常是在测试过程中发现问题后进行, 因此调试总是作为测试的结果发生, 但两者本质上不是同一个过程.

(T) 24. 调试不是测试, 但总是在测试之后发生.

( ) 22-25. Which of the following is an approach to debugging? 309

A. backtracking
B. brute force
C. cause elimination
D. code restructuring

A项backtracking(回溯)、B项brute force(蛮力法)、C项cause elimination(原因消除法)都是常用调试方法: 回溯是从出错点往前查找原因, 蛮力法是逐步插入打印语句或单步执行来查找错误, 原因消除法是通过假设和排除法定位错误. D项code restructuring(代码重构)是提高代码质量和可维护性的方法, 不属于调试方法.

(ABC) 25. 以下哪项是调试的方法?
A. **回溯法**
B. **穷举法**
C. **原因消除法**
D. 代码重构

( ) 23-1. With thorough testing it is possible to remove all defects from a program prior to delivery to the customer. 310

无论测试多么全面, 都无法保证发现和消除所有缺陷. 软件复杂性高, 输入、路径和使用场景可能无限, 测试只能覆盖有限情况. 有些缺陷只在特定环境或极端情况下才会暴露, 因此彻底消除所有缺陷几乎不可能.

(F) 1. 通过彻底的测试, 可以在交付给客户之前从程序中消除所有缺陷.

( ) 23-2. Which of the following are characteristics of testable software? 311

A. observability
B. simplicity
C. stability
D. all of the above

可测试性软件是便于测试、易于发现和定位缺陷的软件. 可观察性(observability)指能清楚看到软件内部状态和输出, 便于判断错误. 简单性(simplicity)表示结构简单、模块清晰, 减少测试复杂度. 稳定性(stability)是指软件修改后行为稳定, 不易引入新错误, 便于重复测试. A、B、C都是可测试性软件的重要特征.

(D) 2. 下列哪些是可测试软件的特征?
A. 可观测性
B. 简单性
C. 稳定性
D. **以上全部**

( ) 23-3. The testing technique that requires devising test cases to demonstrate that each program function is operational is called 312

A. black-box testing
B. glass-box testing
C. grey-box testing
D. white-box testing

题干要求设计测试用例, 证明每个程序功能是否可用, 这属于关注程序功能、只看输入输出、不关心内部实现的测试方法, 即黑盒测试(black-box testing). 黑盒测试主要验证软件功能是否符合需求规格说明书的要求, 不涉及程序内部结构.

(A) 3. 需要设计测试用例以证明每个程序功能都能正常运行的测试技术被称为
A. **黑盒测试**
B. 玻璃盒测试
C. 灰盒测试
D. 白盒测试

( ) 23-4. The testing technique that requires devising test cases to exercise the internal logic of a software module is called 313

A. behavioral testing
B. black-box testing
C. grey-box testing
D. white-box testing

white-box testing(白盒测试)是指测试人员根据软件模块的内部逻辑设计测试用例, 覆盖和验证代码的各个分支、路径和语句, 确保程序内部运行正确. black-box testing(黑盒测试)是按功能需求测试, 不关注内部实现; behavioral testing关注外部行为; grey-box testing结合了黑盒和白盒的特性.

(D) 4. 需要设计测试用例以执行软件模块内部逻辑的测试技术被称为
A. 行为测试
B. 黑盒测试
C. 灰盒测试
D. **白盒测试**

( ) 23-5. What types of errors are missed by black-box testing and can be uncovered by white-box testing? 314

A. behavioral errors
B. logic errors
C. performance errors
D. typographical errors

黑盒测试只关注输入和输出, 无法检测程序内部的逻辑和代码实现, 因此逻辑错误(B)和拼写错误(D)这类只能在代码内部发现的问题, 通常会被黑盒测试遗漏, 但白盒测试可以发现. 行为错误(A)和性能错误(C)通常黑盒测试也能检测.

(BD) 5. 黑盒测试容易遗漏哪些类型的错误, 而这些错误可以通过白盒测试发现?
A. 行为性错误
B. **逻辑错误**
C. 性能错误
D. **拼写错误**

( ) 23-6. Program flow graphs are identical to program flowcharts. 315

程序流程图(flowchart)用图形符号(如矩形、菱形)表示程序步骤和流程控制关系, 主要用于展示程序的整体结构和控制流程. 程序流图(flow graph)更抽象, 用节点表示基本块, 有向边表示控制流, 主要用于程序分析和测试, 如路径覆盖等. 两者都描述程序流程, 但在表达方式、用途和细节上有明显区别, 并不相同.

(F) 6. 程序流程图与程序流程图表是相同的.

( ) 23-7. The cyclomatic complexity metric provides the designer with information regarding the number of 316

A. cycles in the program
B. errors in the program
C. independent logic paths in the program
D. statements in the program

圈复杂度(Cyclomatic Complexity)用于衡量程序的复杂性, 表示程序中独立逻辑路径的数量. 每增加一个判断或分支, 独立路径数就增加. 设计者可以通过圈复杂度判断需要多少测试用例覆盖所有路径.

(C) 7. 圈复杂度度量为设计者提供了关于程序中数量的信息
A. 程序中的环路数
B. 程序中的错误数
C. **程序中独立逻辑路径的数量**
D. 程序中的语句数

( ) 23-8. The cyclomatic complexity of a program can be computed directly from a PDL representation of an algorithm without drawing a program flow graph. 317

圈复杂度(cyclomatic complexity)用于衡量程序逻辑的复杂程度, 通常通过流程图计算. 但只要有PDL(伪代码)描述, 可以分析其中的分支、条件和循环结构, 直接计算圈复杂度, 无需画流程图.

(T) 8. 程序的圈复杂度可以直接通过算法的PDL(程序设计语言)表示计算出来, 而无需绘制程序流程图.

( ) 23-9. Condition testing is a control structure testing technique where the criteria used to design test cases is that they 318

A. rely on basis path testing
B. exercise the logical conditions in a program module
C. select test paths based on the locations and uses of variables
D. focus on testing the validity of loop constructs

条件测试(Condition testing)是一种控制结构测试技术, 设计测试用例时重点在于覆盖和执行程序模块中的各种逻辑条件(如if、while等的判断条件), 确保这些条件能分别取到不同的真假值. 选项B描述了这一点, 其他选项指的是其他测试方法的重点.

(B) 9. 条件测试是一种控制结构测试技术, 其设计测试用例的标准是
A. 依赖于基本路径测试
B. **执行程序模块中的逻辑条件**
C. 根据变量的位置和用途选择测试路径
D. 侧重于测试循环结构的有效性

( ) 23-10. Data flow testing is a control structure testing technique where the criteria used to design test cases is that they 319

A. rely on basis path testing
B. exercise the logical conditions in a program module
C. select test paths based on the locations and uses of variables

D. focus on testing the validity of loop constructs

数据流测试(Data flow testing)是一种结构化测试技术，关注程序中变量的定义、使用和消亡过程．它通过分析变量在不同位置的赋值和使用情况，设计测试用例以发现潜在错误．A描述的是基础路径测试，B是条件覆盖，D是循环测试．

(C) 10. 数据流测试是一种控制结构测试技术，其设计测试用例所依据的标准是
A. 依赖于基本路径测试
B. 执行程序模块中的逻辑条件
C. **根据变量的位置和使用情况选择测试路径**
D. 侧重于测试循环结构的有效性

( ) 23-11. Loop testing is a control structure testing technique where the criteria used to design test cases is that they 320

A. rely basis path testing
B. exercise the logical conditions in a program module
C. select test paths based on the locations and uses of variables
D. focus on testing the validity of loop constructs

Loop testing 是专门针对程序中循环结构的测试方法，设计测试用例时关注各种循环的有效性，如循环0次、1次、多次、最大次数、超出次数等情况．选项D强调"focus on testing the validity of loop constructs"，即关注循环结构的有效性测试，正好对应Loop testing的本质．

(D) 11. 循环测试是一种控制结构测试技术，其设计测试用例的标准是
A. 依赖基本路径测试
B. 执行程序模块中的逻辑条件
C. 根据变量的位置和用途选择测试路径
D. **侧重于测试循环结构的有效性**

( ) 23-12. Black-box testing attempts to find errors in which of the following categories 321

A. incorrect or missing functions
B. interface errors
C. performance errors
D. none of the above

黑盒测试关注软件外部行为，不涉及内部实现．它通过输入与输出测试，发现功能错误(如功能不正确或缺失)、接口错误(如模块间交互问题)、性能错误(如运行速度或资源消耗不达标)等，因此A、B、C都是黑盒测试要发现的错误类型．

(ABC) 12. 黑盒测试试图发现以下哪些类别的错误
A. **不正确或缺失的功能**
B. **接口错误**
C. **性能错误**
D. 以上都不是

( ) 23-13. Graph-based testing methods can only be used for object-oriented systems 322

图形化测试方法通过将程序结构或流程表示为图(如控制流图、数据流图)来设计测试用例，适用于面向过程、面向对象、函数式等各种类型的软件系统，并不局限于面向对象系统，因此具有广泛适用性．

(F) 13. 基于图的测试方法只能用于面向对象系统．

( ) 23-14. Equivalence testing divides the input domain into classes of data from which test cases can be derived to reduce the total number of test cases that must be developed. 323

等价类测试(Equivalence Testing)是一种常用的软件测试方法，其核心思想是将输入数据划分为若干等价类，每个等价类中的数据在程序处理时被认为是等效的．只需从每个等价类中选取代表性数据作为测试用例，可以减少设计和执行的测试用例数量，同时保证测试覆盖性．

(T) 14. 等价类测试将输入域划分为数据类，从这些数据类中可以派生测试用例，以减少必须开发的测试用例总数．

( ) 23-15. Boundary value analysis can only be used to do white-box testing. 324

边界值分析不仅用于白盒测试，也用于黑盒测试，是常见的黑盒测试方法，主要检查输入或输出的边界值是否被正确处理．白盒测试关注代码结构，边界值分析则关注输入输出的极限情况，因此不限于白盒测试．

(F) 15. 边界值分析只能用于白盒测试．

( ) 23-16. Orthogonal array testing enables the test designer to maximize the coverage of the test cases devised for relatively small input domains. 325

正交数组测试(Orthogonal array testing)通过选取具有代表性的测试用例组合，在有限测试用例下尽量覆盖多种输入组合，提升测试效率，适用于输入条件较少但组合较多的情况．

(T) 16. 正交数组测试使测试设计者能够在相对较小的输入域内最大化测试用例的覆盖率．

( ) 23-17. Test derived from behavioral class models should be based on the 326

A. data flow diagram
B. object-relation diagram
C. state transition diagram
D. use-case diagram

行为类模型描述对象在不同状态下根据事件发生状态转移．状态转换图(state transition diagram)用于表示对象的各种状态及状态之间的转换．从行为类模型导出的测试应基于状态转换图，因为它能覆盖对象在不同状态下的行为变化．数据流图、对象关系图和用例图分别描述数据流、对象间关系和系统功能，并不针对行为状态．

(C) 17. 基于行为类模型的测试应基于
A. 数据流图
B. 对象关系图
C. **状态转换图**
D. 用例图

( ) 23-18. Documentation does not need to be tested. 327

文档(如需求文档、用户手册等)需要进行测试和审核，以保证内容准确、完整、易懂，能正确指导用户和开发者．文档出错可能导致软件被误用或开发偏差，因此文档的测试和评审非常重要．

(F) 18. 文档不需要进行测试．

( ) 23-19. Real-time applications add a new and potentially difficult element to the testing mix 328

A. performance
B. reliability
C. security
D. time

实时应用(Real-time applications)的主要特点是对"时间"非常敏感，系统不仅要正确完成功能，还必须在严格的时间限制内响应．因此，测试实时应用时要特别关注时序、响应延迟等时间因素，这与普通应用不同．

(D) 19. 实时应用程序为测试带来了新的且可能较难处理的因素
A. 性能
B. 可靠性
C. 安全性
D. **时间**

( ) 24-1. It is not possible to test object-oriented software without including error discovery techniques applied to the system OOA and OOD models. 329

测试面向对象软件时，必须对面向对象分析(OOA)和面向对象设计(OOD)模型应用错误发现技术．OOA和OOD模型定义了系统的结构和行为，如类、对象、继承和消息传递等．如果这些模型存在错误且未被发现，即使代码测试做得再充分，也可能遗漏系统性问题．因此，只有在测试阶段对这些模型进行错误发现，才能保证系统设计和分析阶段没有遗漏或错误，否则后续测试就不完整，无法有效保证软件质量．

(T) 1. 如果不将错误发现技术应用于系统的面向对象分析(OOA)和面向对象设计(OOD)模型，就无法对面向对象软件进行测试．

( ) 24-2. The correctness of the OOA and OOD model is accomplished using formal technical reviews by the software quality assurance team. 330

正式技术评审能帮助发现OOA和OOD模型中的问题，但不能完全保证模型的正确性．模型的正确性还需要通过模型验证、测试、建模工具自动检查等其他手段．质量保证团队的评审只是保证正确性的手段之一，而不是唯一或全部的方法．

(F) 2. OOA 和 OOD 模型的正确性是通过软件质量保证团队的正式技术评审来实现的．

( ) 24-3. The consistency of object-oriented models may be judged by reviewing the CRC card model. 331

CRC卡(Class-Responsibility-Collaborator卡片)用于描述类、其责任及与其他类的协作关系．检查CRC卡模型可以判断类的职责分配是否合理、协作关系是否清晰，从而评估面向对象模型的一致性．

(T) 3. 可以通过审查CRC卡模型来判断面向对象模型的一致性．

( ) 24-4. Test case design for OO software is driven by the algorithmic detail of the individual operations. 332

面向对象(OO)软件的测试用例设计需要关注每个操作(类的方法)的算法细节．每个操作内部的逻辑决定了它在不同输入下的行为，测试用例要覆盖这些逻辑分支和可能的执行路径，以确保操作的正确性．因此，测试用例设计是由各操作的算法细节驱动的．

(T) 4. 面向对象软件的测试用例设计是由各个操作的算法细节驱动的．

( ) 24-5. Integration testing of object-oriented software can be accomplished by which of the following testing strategies? 333

A. Cluster testing
B. Glass-box testing
C. Thread-based testing
D. Use-based testing

Cluster testing(簇测试)、Thread-based testing(基于线程的测试)、Use-based testing(基于用例的测试)都是面向对象集成测试常用的方法．Cluster testing是将相关类组合成簇进行测试；Thread-based testing沿系统某个功能流程(线程)测试相关类的协作；Use-based testing根据系统用例来集成和测试对象．Glass-box testing(玻璃盒测试)是白盒测试方法，主要用于单元测试阶段，不是专门针对集成测试的策略．

(ACD) 5. 面向对象软件的集成测试可以通过以下哪些测试策略来完成?
A. **簇测试**
B. 玻璃盒测试
C. **基于线程的测试**
D. **基于使用的测试**

( ) 24-6. Validation of object-oriented software focuses on user visible actions and outputs from the system. 334

面向对象软件的验证(Validation)主要关注用户可见的系统动作和输出．验证强调"做正确的事"，即系统是否满足用户需求和期望，是否按照需求规格说明书正确实现功能．对于面向对象的软件，验证同样关注用户界面、功能输出等用户可感知的部分，而不是只关注内部结构或实现细节．

(T) 6. 面向对象软件的验证侧重于用户可见的操作和系统输出．

( ) 24-7. Encapsulation of attributes and operations inside objects makes it easy to obtain object state information during testing. 335

"封装"是面向对象编程的重要特性，将对象的属性(数据)和操作(方法)隐藏在对象内部，只能通过特定方法访问．这样做提高了安全性和模块化，但在测试时，获取对象的内部状态信息变得不容易，通常无法直接访问私有属性．因此，封装实际上增加了测试时获取对象状态信息的难度．

(F) 7. 将属性和操作封装在对象内部，使得在测试过程中容易获取对象的状态信息.

( ) 24-8. Use-cases can provide useful input into the design of black-box and state-based tests of OO software. 336

用例(use-cases)描述了用户与系统的交互,明确了系统应实现的功能和行为. 黑盒测试关注功能是否按需求实现，可以根据用例设计测试场景来验证功能. 状态测试关注对象在不同状态下的行为变化，用例中的情景有助于识别和覆盖不同的状态转换，因此用例是设计这两类测试的重要依据.

(T) 8. 用例可以为面向对象软件的黑盒测试和基于状态的测试的设计提供有用的输入.

( ) 24-9. Fault-based testing is best reserved for 337

A. conventional software testing
B. operations and classes that are critical or suspect
C. use-case validation
D. white-box testing of operator algorithms

Fault-based testing(基于错误的测试)主要针对系统中可能存在的特定故障或错误，注重容易出错、风险高的部分. 它适用于关键或可疑的操作和类, 有助于发现潜在缺陷, 提高软件质量. 选项B"operations and classes that are critical or suspect"正好描述了这种场景.

(B) 9. 基于故障的测试最适合用于
A. 传统软件测试
B. **关键或可疑的操作和类**
C. 用例验证
D. 操作员算法的白盒测试

( ) 24-10. Scenario-based testing 338

A. concentrates on actor and software interaction
B. misses errors in specifications
C. misses errors in subsystem interactions
D. both a and b

场景驱动测试(Scenario-based testing)主要关注用户(actor)和软件之间的交互，即系统在真实使用场景下的表现. 它不专注于发现规范中的错误或各子系统之间的交互问题, 因此只关注用户和软件的交互.

(A) 10. 基于场景的测试
A. **侧重于参与者与软件的交互**
B. 遗漏规范中的错误
C. 遗漏子系统交互中的错误
D. a 和 b 都是

( ) 24-11. Random order tests are conducted to exercise different class instance life histories. 339

随机顺序测试(Random order tests)是通过随机创建、使用和销毁类实例，覆盖更多对象状态和生命周期路径，帮助发现潜在错误. 在面向对象软件测试中，这种方法有助于检验不同类实例的生命周期历史.

(T) 11. 随机顺序测试用于检验不同类实例的生命周期.

( ) 24-12. Which of these techniques is not useful for partition testing at the class level 340

A. attribute-based partitioning
B. category-based partitioning
C. equivalence class partitioning
D. state-based partitioning

类级别的分区测试通常采用属性、类别和状态为基础的划分方法, 更贴合面向对象设计中类的特性. 等价类划分一般用于函数级别，将输入分为有效和无效等价类，而不是针对类的属性或状态进行，因此不适用于类级别的分区测试.

(C) 12. 以下哪种技术对于类级别的划分测试没有用处?
A. 基于属性的划分
B. 基于类别的划分
C. **等价类划分**
D. 基于状态的划分

( ) 24-13. Multiple class testing is too complex to be tested using random test cases. 341

多类测试(multiple class testing)虽然复杂，但仍可以用随机测试用例来测试. 随机测试用例在多类测试中可以帮助发现潜在问题，只是覆盖率和针对性可能不如专门设计的测试用例高. 复杂性不会完全阻止使用随机方法.

(F) 13. 多类测试过于复杂，无法使用随机测试用例进行测试.

( ) 24-14. The state model can be used to derive test cases based on the dynamic behavior of an object-oriented system. 342

状态模型(state model)描述了对象在不同状态间的转换过程，展示对象如何响应各种事件和操作. 通过分析状态模型，可以设计测试用例，覆盖对象的各种状态变化和动态行为，确保系统在不同情况下正常工作. 因此，状态模型可以用于派生基于动态行为的测试用例.

(T) 14. 状态模型可以用于根据面向对象系统的动态行为派生测试用例.

( ) 25-1. Which of the following is not one of the dimensions of quality used to assess a WebApp? 343

A. Content
B. Maintainability
C. Navigability
D. Usability

内容(Content)、可导航性(Navigability)、可用性(Usability)是评估Web应用质量常用的维度，主要关注用户体验和信息呈现. 可维护性(Maintainability)虽然在软件工程中很重要, 但更关注代码和后期维护，不是评价Web应用用户体验和展示质量的常规维度.

(B) 1. 下列哪一项不是用于评估Web应用质量的维度?
A. 内容
B. **可维护性**
C. 可导航性
D. 可用性

( ) 25-2. WebApps require special testing methodologies because WebApp errors have several unique characteristics. 344

Web应用程序需要特别的测试方法, 因为WebApp的错误具有独特特点, 如需支持多种浏览器和设备, 用户量大且分布广，更新频繁，易受安全威胁，网络环境复杂. 这导致传统软件测试方法难以完全满足WebApp需求，因此需要专门的测试方法来应对这些特殊性.

(T) 2. Web应用程序需要特殊的测试方法，因为Web应用程序的错误具有一些独特的特性.

( ) 25-3. Since WebnApps evolve continuously, the testing process is an on-going activity, conducted by the Web support staff using regression tests. 345

Web应用会持续演化，经常更新内容、改进功能或修复bug, 这些变化可能带来新问题. 为了确保每次修改后系统仍能正常工作, 测试需要持续进行. 回归测试是在系统修改后反复验证原有功能是否正常，这项工作一般由Web支持团队负责，因此测试是持续的活动.

(T) 3. 由于Web应用程序持续演化，测试过程是一项持续进行的活动，由Web支持人员通过回归测试来执行.

( ) 25-4. Test planning is not used in WebApp testing. 346

题目说"测试计划在Web应用测试中不被使用"，这是错误的. 无论是传统软件还是Web应用，测试计划都是非常重要的一步. 测试计划帮助团队明确测试目标、范围、方法、时间安排等，保证测试过程有序进行. 所以Web应用测试同样需要测试计划.

(F) 4. 测试计划在Web应用测试中不被使用.

( ) 25-5. As the WebApp architecture is constructed which types of testing are used as integration tests? 347

A. Component testing
B. Content testing
C. Navigation testing
D. Usability testing

集成测试主要验证模块、组件组合后的协作. Component testing(组件测试)检查模块间接口和交互是否正确，Navigation testing(导航测试)验证页面或模块间链接是否顺畅、逻辑是否正确，这两者属于集成测试. Content testing(内容测试)和Usability testing(可用性测试)侧重内容本身或用户体验，不是集成测试的主要方法.

(AC) 5. 当WebApp架构被构建时，哪些类型的测试被用作集成测试?
A. **构件测试**
B. 内容测试
C. **导航测试**
D. 可用性测试

( ) 25-6. Which of the following is not one of the objectives of WebApp content testing? 348

A. Find organizational or structure errors
B. Identify linking errors
C. Uncover semantic errors
D. Uncover syntactic errors

内容测试主要关注内容本身的正确性, 包括内容结构(A)、语义(C)、语法(D)等. B选项"Identify linking errors"指的是链接错误，属于导航或界面测试，不属于内容测试的目标.

(B) 6. 以下哪一项不是Web应用内容测试的目标?
A. 发现组织或结构错误
B. **识别链接错误**
C. 发现语义错误
D. 发现语法错误

( ) 25-7. Database testing is very rarely a part of WebApp content testing. 349

Web应用的内容通常来自数据库，如用户信息、商品数据等. 为了保证页面展示的数据准确，内容测试时常常需要检查数据库数据的准确性、完整性，以及能否正确读取和显示. 因此，数据库测试是WebApp内容测试的重要部分，并非很少涉及.

(F) 7. 数据库测试很少作为Web应用内容测试的一部分.

( ) 25-8. The overall strategy for interface testing is to uncover errors 350

A. in navigation semantics
B. in overall usabililty
C. related to specific interface mechanisms
D. both a and c

接口测试(interface testing)的主要目标是发现界面中的错误，包括导航语义方面的错误，即用户在界面中跳转、操作时，是否能达到预期的功能和效果，是否存在跳转或逻辑错误，以及与具体界面机制相关的错误，比如按钮、菜单、输入框等组件的功能实现是否正确. B选项中的整体可用性主要属于用户体验测试，不是接口测试的主要目标.

(D) 8. 界面测试的总体策略是发现错误
A. 在导航语义中
B. 在整体可用性中
C. 与特定界面机制相关
D. **a 和 c 都是**

( ) 25-9. Which of the following is not a WebApp interface mechanism? 351

A. Browser
B. Cookies
C. Forms
D. Links

本题考查Web应用(WebApp)的接口机制. WebApp接口机制是用户与Web应用交互的方法或工具. A. Browser(浏览器)是用户访问和操作WebApp的基本工具，属于接口机制. B. Cookies(Cookie)是WebApp用于存储和管理用户信息的机制, 也属于接口机制. C. Forms(表单)是用户输入信息的页面元

素, 不是接口机制, 而是界面组件. D. Links(链接)用于网页间导航, 也属于接口机制. C选项"Forms"不是WebApp的接口机制, 而是页面的组成部分.

(C) 9. 以下哪一项不是Web应用的界面机制?
A. 浏览器
B. Cookies(浏览器缓存)
C. **表单**
D. 链接

( ) 25-10. When testing WebApp interface semantics, each use-case is used as input for the design of a testing sequence. 352

Web应用接口语义测试时, 用例(use-case)描述了用户与系统交互的具体场景. 在测试WebApp接口时, 每个用例可作为设计测试序列的输入. 用例帮助确定测试的步骤和数据, 用于检验界面是否按预期处理各种用户操作.

(T) 10. 在测试Web应用程序界面语义时, 每个用例都被用作测试序列设计的输入.

( ) 25-11. Usability tests should be designed and executed by intended users for a given WebApp. 353

可用性测试(Usability tests)应由专业测试人员或开发团队负责设计和执行, 目标用户只是参与测试, 提出反馈, 并不负责设计和实施测试. 这样能保证测试的科学性和全面性.

(F) 11. 可用性测试应该由特定Web应用的预期用户来设计和执行.

( ) 25-12. WebApp compatibility testing is conducted to be sure that the user model for usage scenario matched the user category assigned to a given user. 354

兼容性测试(compatibility testing)主要确保Web应用能在不同浏览器、操作系统、设备等环境下正常运行, 关注技术层面的兼容性, 不涉及用户模型和用户类别的匹配. 用户模型和使用场景的匹配属于可用性测试或需求分析的范畴, 不是兼容性测试的目标. 因此, 这句话是错误的.

(F) 12. WebApp兼容性测试的目的是确保使用场景下的用户模型与分配给特定用户的用户类别相匹配.

( ) 25-13. Which test case design technique(s) are appropriate for WebApp component-level testing? 355

A. Boundary value analysis
B. Equivalence partitioning
C. Path testing
D. All of the above

Web应用组件级测试常用的测试用例设计技术包括边界值分析(Boundary value analysis)、等价类划分(Equivalence partitioning)和路径测试(Path testing). 边界值分析和等价类划分主要用于检查输入数据的有效性, 路径测试用于检查程序的逻辑流程. WebApp组件通常涉及输入处理和逻辑流程, 因此这三种方法都适用.

(D) 13. 哪些测试用例设计技术适用于WebApp组件级测试?
A. 边界值分析
B. 等价类划分
C. 路径测试
D. **以上全部**

( ) 25-14. The purpose of WebApp navigation syntactic testing is to ensure the correct appearance of each navigation mechanism. 356

WebApp导航语法测试(navigation syntactic testing)主要关注导航机制是否按照设计正确实现, 如链接是否有效、按钮是否能跳转到正确页面等. 它测试的是导航的功能和行为, 不涉及每个导航机制的外观(appearance). 外观属于界面测试(UI testing)的范畴.

(F) 14. Web应用导航语法测试的目的是确保每个导航机制的正确外观.

( ) 25-15. Both Web engineers and non-technical users conduct navigation semantics testing for WebApps. 357

Web工程师和非技术用户都会参与Web应用的导航语义测试. 导航语义测试主要检查网站导航的合理性和易用性, 确保用户能顺利找到需要的信息. Web工程师从技术和结构角度测试, 非技术用户则从实际使用角度发现问题. 两者共同参与, 有助于提升Web应用的技术质量和用户体验.

(T) 15. Web工程师和非技术用户都会对Web应用进行导航语义测试.

( ) 25-16. Which of following is not one of the elements that need to be considered when constructing WebApp server-side configuration tests? 358

A. Browser compatibility
B. Database software integration
C. Operating system compatibility
D. System security measures

服务器端配置测试关注服务器的环境和相关集成, 如数据库软件的集成(B)、操作系统兼容性(C)、系统安全措施(D)等. 浏览器兼容性(A)属于客户端的考虑内容, 不包括在服务器端配置测试中.

(A) 16. 下列哪一项不是构建WebApp服务器端配置测试时需要考虑的要素?
A. **浏览器兼容性**
B. 数据库软件集成
C. 操作系统兼容性
D. 系统安全措施

( ) 25-17. To design client-side configuration tests each user category is assessed to reduce the number of configuration variables to a manageable number. 359

在设计客户端配置测试时, 会对每种用户类别进行评估, 以减少配置变量的数量, 使其变得可管理. 如果不对用户进行分类, 配置变量的组合会非常多, 难以测试. 通过对用户分类, 可以有针对性地选择有代表性的配置, 减少测试工作量, 提高测试效率. 这是实际测试中常用的方法.

(T) 17. 为了设计客户端配置测试, 需要评估每个用户类别, 以将配置变量的数量减少到可管理的范围.

( ) 25-18. Which of the following is not a testable WebApp security element? 360

A. Authentication
B. Encryption
C. Firewalls
D. Penetration

D项Penetration(渗透)不是Web应用程序可以直接测试的安全元素. A(认证)、B(加密)、C(防火墙)都是具体的安全机制, 可以通过测试验证其有效性. D(渗透)指的是渗透测试, 是一种测试方法, 不是具体的安全机制或元素, 因此不能像认证、加密、防火墙那样被直接测试.

(D) 18. 下列哪一项不是可测试的Web应用安全要素?
A. 认证
B. 加密
C. 防火墙
D. **渗透**

( ) 25-19. WebApp performance tests are designed to 361

A. asses WebApp usability
B. evaluate page loading times
C. simulate real-world loading situations
D. test network connectivity

WebApp性能测试的主要目的是模拟真实世界中的负载情况, 评估Web应用在不同用户数量和使用场景下的表现. A是可用性测试内容, B只是性能测试的一个方面, D是网络连接测试, 不属于WebApp本身的性能测试.

(C) 19. WebApp 性能测试的目的是
A. 评估 WebApp 的可用性
B. 评估页面加载时间
C. **模拟真实世界的加载情况**
D. 测试网络连接性

( ) 25-20. Load testing involves determining the input of which 3 variables? 362

A. N, T, D
B. N, T, P
C. T, D, P
D. N, D, P

负载测试需要确定三个变量: N是并发用户数(Number of users), T是事务数(Number of transactions), D是持续时间(Duration). 负载测试时, 会设定有多少用户同时操作(N)、每个用户执行多少事务(T)、测试持续多长时间(D). 选项A(N, T, D)对应这三项.

(A) 20. 负载测试涉及确定哪三个变量的输入?
A. **N、T、D**
B. N、T、P
C. T、D、P
D. N、D、P

( ) 25-21. WebApp stress testing is a continuation load testing. 363

负载测试(load testing)关注系统在正常或高负载下的表现, 验证其能否正常运行. 压力测试(stress testing)是在负载测试基础上, 继续增加负载, 直到系统崩溃或达到极限, 以观察系统的极限稳定性和恢复能力. 因此, 压力测试可视为负载测试的进一步或延续.

(T) 21. WebApp压力测试是负载测试的延续.

( ) 26-1. MobileApps require special testing methodologies because of concerns associated using them in diverse network environments. 364

移动应用通常在各种不同的网络环境下运行, 如Wi-Fi、4G、5G或网络不稳定时. 不同网络状况会影响应用的性能、响应速度和功能稳定性, 因此需要专门的测试方法, 以确保应用在各种网络条件下都能正常工作. 这也是移动应用测试比传统桌面软件测试更复杂的原因之一.

(T) 1. 移动应用程序需要特殊的测试方法, 因为在不同的网络环境中使用它们会带来相关问题.

( ) 26-2. Since MobileApp users are attracted to new technologies they are very tolerant of errors and testing effort can be reduced. 365

虽然移动应用用户喜欢新技术, 但他们对应用的质量和稳定性要求很高. 如果出现错误或崩溃, 用户很容易卸载应用或给出差评. 因此, 不能因为用户喜欢新技术就减少测试工作, 移动应用需要充分测试以保证良好的用户体验.

(F) 2. 由于MobileApp用户对新技术很感兴趣, 因此他们对错误非常宽容, 可以减少测试工作量.

( ) 26-3. Designing test cases directly from user stories increase the likelihood of developing effective test cases in a timely manner. 366

直接从用户故事设计测试用例, 可以保证测试内容符合实际需求, 帮助发现需求中的问题, 提高测试的针对性和有效性. 用户故事简洁明了, 便于开发和测试团队快速理解需求, 减少沟通成本, 更及时地编写有效的测试用例.

(T) 3. 直接根据用户故事设计测试用例可以提高及时开发有效测试用例的可能性.

( ) 26-4. Automated testing tools eliminate the need to do regression testing for MobileApps. 367

自动化测试工具不能消除对移动应用回归测试的需求. 它们可以提高回归测试的效率, 但无法完全代替回归测试. 每次软件修改后, 都需要验证旧功能是否正常, 自动化只是一种手段, 回归测试仍然必不可少.

(F) 4. 自动化测试工具可以消除对移动应用进行回归测试的需求.

( ) 26-5. A weighted device platform matrix helps to prioritize test cases. 368

加权设备平台矩阵(weighted device platform matrix)用于评估和选择在不同设备或平台上测试的优先级，主要目的是确定要支持哪些设备或平台．它不是用来直接对测试用例进行优先级排序．测试用例的优先级通常与需求的重要性、风险等因素相关，而不是依赖于设备平台矩阵．

(F) 5. 加权的设备平台矩阵有助于对测试用例进行优先级排序．

(　) 26-6. Part of the reason for stress testing is to ensure that the MobileApp exhibits graceful degradation on failure. 369

压力测试的目的不仅是找出系统在高负载下的崩溃点，更重要的是观察系统在超出极限后是否能保持基本功能或以合理方式提示用户，而不是直接崩溃．优雅降级指系统在异常或资源不足时，尽量保持核心功能或给出友好的错误信息，提高用户体验和系统健壮性．

(T) 6. 压力测试的部分原因是为了确保MobileApp在发生故障时能够优雅降级．

(　) 26-7. Which of the following are reasons for testing in the wild? 370

A. Assessing the impact of production environments
B. Failing to create test cases
C. Not understanding user demographics
D. Testing for variable performance on user devices

“野外测试”(testing in the wild)的目的是评估生产环境的影响(A)和测试用户设备上的性能变化(D)，因为实验室环境无法完全模拟实际用户的使用场景和设备差异．B和C属于测试准备或需求分析阶段的问题，不是野外测试的理由．

(AD) 7. 以下哪些是进行野外测试(实际环境测试)的原因?
A. **评估生产环境的影响**
B. 未能创建测试用例
C. 不了解用户群体特征
D. **测试用户设备上性能的变化**

(　) 26-8. When testing the quality of user interaction the focus should be on user visible interaction mechanisms. 371

在测试用户交互质量时，应关注用户可见的交互机制．用户交互的质量直接影响用户体验和满意度．测试时主要关注用户界面、按钮、菜单、输入输出等用户能直接看到和操作的部分，确保这些交互机制直观、易用、响应正常．

(T) 8. 在测试用户交互的质量时，重点应放在用户可见的交互机制上．

(　) 26-9. Which of that following add to the difficulty of testing MobileApp gestures? 372

A. Automatic tool use is difficult
B. Creating functions to simulate events
C. Screen size variation
D. Using paper prototypes

A选项“自动化工具的使用很困难”，因为现有的自动化测试工具对复杂手势支持有限．B选项“创建模拟事件的函数”增加了难度，要精确模拟各种手势事件比较难．C选项“屏幕尺寸的变化”也是重要因素，不同设备的屏幕大小会影响手势的识别和表现．D选项“使用纸面原型”不会影响软件测试本身，因为纸面原型主要用于设计阶段，不涉及实际的手势测试．

(ABC) 9. 下列哪些因素增加了测试移动应用手势的难度?
A. **自动化工具的使用较为困难**
B. **创建用于模拟事件的函数**
C. **屏幕尺寸的多样性**
D. 使用纸质原型

(　) 26-10. Continuous speech recognition techniques have eliminated the need for key entry in MobileApps. 373

连续语音识别技术可以让用户通过语音与应用交互，减少了对键盘的依赖，但没有完全消除键盘输入的需求．语音识别会受到环境噪音、口音差异、识别准确率等因素影响，很多场景下用户仍然需要通过键盘输入来保证准确性或完成特定操作．

(F) 10. 连续语音识别技术已经消除了在移动应用中键盘输入的需求．

(　) 26-11. Predictive technologies are often used to help speed up virtual keyboard inpu on mobile devices. 374

预测技术常用于提升移动设备虚拟键盘的输入速度．由于屏幕小、手指操作有限，预测技术(如词语联想、自动补全)能根据已输入内容预测下一个可能的字词，减少按键次数，提高输入效率．这类技术在现代手机、平板等设备上十分常见．

(T) 11. 预测技术通常用于帮助加快移动设备上虚拟键盘的输入速度．

(　) 26-12. The ability of a MobileApp to handle alerts without disrupting user workflow must be tested in the production environment? 375

移动应用在不打断用户操作的情况下处理警报，属于用户体验和功能性测试．这类测试通常应在开发和测试环境中完成，因为生产环境主要用于正式发布和实际运行．只有测试环境验证通过后，才会部署到生产环境．直接在生产环境测试有风险，可能影响真实用户，所以不需要在生产环境进行这种测试．

(T) 12. 必须在生产环境中测试移动应用处理警报而不干扰用户工作流程的能力吗?

(　) 26-13. The Testing across borders is not necessary each MobileApp is developed for use in a specific country. 376

题目认为每个移动应用都是为特定国家开发的，所以不需要跨国测试．实际上，即使应用最初只面向某国，用户也可能在其他国家使用，或未来可能进入国际市场．不同国家在语言、法律、网络环境、硬件设备等方面存在差异，因此跨国测试是必要的，以确保应用能在不同环境下正常运行．

(F) 13. 跨国测试不是必需的，因为每个移动应用程序都是为特定国家开发的．

(　) 26-14. Which of the following are issues that make real-time testing difficult? 377

A. Limited device processing capacity
B. Power limitations on the device
C. Unique mobile network infrastructures
D. All of the above

实时测试要求系统在严格时间限制内响应．在移动或嵌入式设备上，会遇到设备处理能力有限(A)、电源受限(B)、各地移动网络基础设施差异(C)等问题，这些都会影响系统响应速度和测试准确性．

(D) 14. 以下哪些问题使实时测试变得困难?
A. 设备处理能力有限
B. 设备的电源限制
C. 独特的移动网络基础设施
D. **以上全部**

(　) 26-15. Device emulators eliminate the need to test MobileApps on actual devices. 378

设备模拟器能在电脑上模拟不同移动设备环境，但无法完全模拟真实设备的硬件特性、性能、传感器、网络状况等．许多兼容性、性能、交互和真实场景下的问题只能在实际设备上发现，所以不能只依赖模拟器，还需要在真实设备上测试．

(F) 15. 设备模拟器可以消除在真实设备上测试移动应用的需求．

(　) 27-1. When analyzing security requirements focus in system assets with the highest value and greatest exposure. 379

在分析安全需求时，应重点关注最有价值、暴露程度最高的系统资产．这些资产被攻击或泄露会造成最大损失，因此安全工作要优先保护这些关键目标．这体现了风险优先原则在安全需求分析中的应用．

(T) 1. 在分析安全需求时，应关注系统中价值最高且暴露程度最大的资产．

(　) 27-2. It is possible to have a safe system that is not secure. 380

安全(safety)和安全性(security)在软件工程中是两个不同的概念．系统可以在正常和故障情况下避免意外伤害，称为有安全性(safety)；但如果系统能被恶意攻击者利用或破坏，缺乏安全性(security)，则不能称为真正的安全系统．没有安全性的系统不能算作安全，因此题目说法错误．

(F) 2. 有可能拥有一个安全但不保密的系统．

(　) 27-3. Individuals rarely expose their personal information to others on social media networks. 381

题目说个人很少在社交媒体上公开个人信息，实际上很多人会主动分享姓名、照片、位置、生日等，这种行为很普遍，并不是“很少”发生．

(F) 3. 个人很少在社交媒体网络上向他人公开他们的个人信息．

(　) 27-4. Wireless networks require the trust and cooperation between nodes that can be exploited by malicious programs? 382

无线网络中的节点(如手机、路由器)通常需要协作完成数据转发或资源共享，这建立在节点间的信任基础上．如果有恶意节点加入，可能会利用这种信任进行攻击，如窃听、篡改数据或拒绝服务等．因此，节点间的信任和协作确实可能被恶意程序利用，带来安全风险．

(T) 4. 无线网络需要节点之间的信任与合作，这可能会被恶意程序利用吗?

(　) 27-5. Cloud computing is has greater levels of security that other web data repositories. 383

云计算的安全性不一定比其他网络数据存储库更高．它取决于云服务提供商的安全措施、用户的安全管理、数据加密等多种因素，而不是天然更安全．因此，不能笼统地认为云计算有更高的安全级别．

(F) 5. 云计算比其他网络数据存储库具有更高的安全性．

(　) 27-6. The security concerns remain an obstacle to implementing the vision implied by the Internet of Things . 384

安全问题一直是实现物联网愿景的障碍．物联网设备会收集和传输大量敏感数据，但这些设备普遍安全性不足，容易被攻击或滥用，进而导致数据泄露和隐私侵犯等问题．因此，安全问题是物联网发展必须面对和解决的重要障碍．

(T) 6. 安全问题仍然是实现物联网愿景的障碍．

(　) 27-7. Security and usability requirements are often in conflict with each other. 385

安全性要求通常需要对系统访问进行限制，如设置复杂密码、多重身份验证等，这些措施虽然提升了安全性，但会使用户操作更复杂，降低易用性．因此，安全性和可用性在实际开发中经常需要权衡和平衡．

(T) 7. 安全性需求和可用性需求通常彼此冲突．

(　) 27-8. Which of following is not one of the elements of a security model? 386

A. Criminal background checks
B. External interface requirements
C. Rules of operation
D. Security policy objectives

安全模型通常包括安全政策目标(D)、操作规则(C)和外部接口要求(B)，用于定义和控制系统的安全保障方式．A选项“犯罪背景调查”属于人员管理或人力资源措施，不属于安全模型的构成要素．

(A) 8. 以下哪项不是安全模型的要素?
A. **犯罪背景调查**
B. 外部接口需求
C. 操作规则
D. 安全策略目标

(　) 27-9. Security metrics and measures need to assess which of these properties? 387

A. Dependability
B. Survivability
C. Trustworthiness
D. All of the above

安全性度量(security metrics and measures)应评估系统的可靠性(Dependability)、生存能力(Survivability)和可信任性(Trustworthiness)．同时评估这些属性，才能全面衡量系统的安全水平．

(D) 9. 安全度量和测量需要评估以下哪些属性?
A. 可靠性
B. 生存能力
C. 可信度
D. **以上全部**

( ) 27-10. Security correctness checks should be included which of the following activities? 388

A. Audits
B. Deployment
C. Inspections
D. Testing

安全性正确性检查是指在不同阶段对软件安全性进行评估和验证. A. Audits(审计)用于核查系统安全要求是否得到满足. B. Deployment(部署)阶段需要进行安全检查, 确保环境和配置不会引入新的安全风险. C. Inspections(评审)通过代码和设计评审发现潜在安全漏洞. D. Testing(测试)虽然安全测试很重要, 但主要是缺陷发现, 不属于检查活动. 因此, 安全性正确性检查应包括A、B、C三项.

(ABC) 10. 安全性正确性检查应包含在以下哪些活动中?
A. **审计**
B. **部署**
C. **评审**
D. 测试

( ) 27-11. Which is not one of the elements of a security case? 389

A. Arguments
B. Bug reports
C. Claims
D. Evidence

安全性案例(security case)包括三大要素: Claims(主张或断言)、Arguments(论证或理由)、Evidence(证据), 它们共同说明系统的安全性. Bug reports(缺陷报告)只是记录软件缺陷的文档, 不属于安全性案例的核心组成部分.

(B) 11. 以下哪项不是安全案例的要素?
A. 论据
B. **缺陷报告**
C. 主张
D. 证据

( ) 27-12. Security assurance and risk identification must be included in the schedule and budget if they are to be taken seriously. 390

安全保障(security assurance)和风险识别(risk identification)如果没有明确列入项目的时间安排和预算, 往往会被忽略或临时处理, 这可能导致后期出现严重的安全问题或风险无法控制. 将它们正式写入进度和预算, 可以确保有足够资源和重视来进行这些工作, 从而保障项目的安全和可控.

(T) 12. 如果要认真对待安全保障和风险识别, 必须将其纳入进度安排和预算.

( ) 27-13. Threat analysis is not needed for conventional software applications. 391

无论是传统软件还是新型软件, 都可能面临安全威胁, 如数据泄露、未授权访问等. 威胁分析有助于发现和预防这些安全风险, 提高软件的安全性和可靠性. 因此, 传统软件应用也需要进行威胁分析.

(F) 13. 对于常规软件应用程序, 不需要进行威胁分析.

( ) 27-14. An incident response plan spells out the actions to be carried out by each stakeholder in response to specific attacks. 392

incident response plan(事件响应计划)会明确规定在发生特定攻击时, 各相关人员(stakeholder)应采取的具体行动. 这有助于组织快速、有序地应对安全事件, 减少损失并恢复正常运行.

(T) 14. 事件响应计划明确规定了每个相关方在应对特定攻击时应采取的行动.

( ) 28-1. The cleanroom strategy is based on the _____ software process model. 393

A. evolutionary
B. incremental
C. revolutionary
D. spiral

cleanroom策略强调高可靠性和缺陷预防, 基于增量(incremental)软件过程模型. 将系统分为多个增量, 每个增量都进行规范、设计、形式化验证和测试, 逐步构建完整系统.

(B) 1. Cleanroom策略基于_____软件过程模型.
A. 演化型
B. **增量型**
C. 革命型
D. 螺旋型

( ) 28-2. The cleanroom strategy relies on 394

A. exhaustive testing
B. extensive unit testing of all modules
C. tests that exercise the software as it is really used
D. white box testing strategies

Cleanroom策略通过形式化设计和严格评审来减少缺陷, 主要依靠统计测试, 即模拟软件在真实环境中的使用方式, 测试系统的可靠性. 因此, Cleanroom方法依赖于能够模拟软件实际使用情况的测试用例. 其他选项关注传统的详细单元测试或代码覆盖, 这些不是Cleanroom的核心.

(C) 2. Cleanroom策略依赖于
A. 穷举测试
B. 对所有模块进行广泛的单元测试
C. **在软件实际使用时进行的测试**
D. 白盒测试策略

( ) 28-3. Use of formal program correctness proofs as part of the cleanroom process eliminates the need do any testing for software defects. 395

即使在cleanroom过程中采用形式化的程序正确性证明, 也不能完全替代测试. 形式化证明只能确保软件与规范一致, 但实现、规范本身或环境中仍可能存在缺陷或遗漏. 因此, 开发中仍需通过测试发现和修正潜在的软件缺陷.

(F) 3. 在洁净室过程(Cleanroom Process)中使用形式化程序正确性证明可以消除对软件缺陷进行任何测试的需要.

( ) 28-4. In cleanroom software engineering a "box" encapsulates some system aspect at a particular level of detail. 396

在Cleanroom方法中, "box structure specification"用不同层次的"盒子"描述系统功能, 每个盒子代表系统的一个部分或功能, 并在不同抽象级别上逐步细化. "box"确实是在特定层次上封装系统内容的.

(T) 4. 在洁净室软件工程中, "盒子"在特定的细节层次上封装了系统的某个方面.

( ) 28-5. This box specification describes an abstraction, stimuli, and response. 397

A. black box
B. clear box
C. state box
D. white box

题目中的"abstraction, stimuli, and response"指只关注输入(stimuli)和输出(response), 不涉及内部实现细节, 这就是黑盒(black box)测试的特点. 黑盒方法关注系统对输入的响应, 不检查内部结构或工作机制.

(A) 5. 该盒子规范描述了抽象、刺激和响应.
A. **黑盒**
B. 透明盒
C. 状态盒
D. 白盒

( ) 28-6. This box specification describes the architectural design for some system component. 398

A. black box
B. clear box
C. state box
D. white box

本题考查不同类型的盒图在软件体系结构设计中的作用. 题干提到的"box specification"用于描述系统组件的体系结构设计. black box(黑盒)只描述输入和输出, 不涉及内部结构; clear box(清盒)不是常见术语; state box(状态盒)在Jackson系统开发方法中用于描述系统组件的结构和状态变化, 适合表达体系结构设计; white box(白盒)强调内部结构和实现细节, 常用于详细设计或编码阶段. 描述体系结构设计的是 state box.

(C) 6. 该盒子规范描述了某个系统组件的体系结构设计.
A. 黑盒
B. 明盒
C. **状态盒**
D. 白盒

( ) 28-7. This box specification is closely aligned with procedural design and structured programming. 399

A. black box
B. clear box
C. state box
D. white box

clear box(透明盒)规格强调对系统内部过程和数据结构的详细描述, 适合过程化设计(procedural design)和结构化程序设计(structured programming), 因为这些方法需要清楚了解程序内部实现细节. black box只关注输入输出, white box类似但更注重逻辑路径, state box关注状态变化, 只有clear box最符合题干描述.

(B) 7. 该盒子规范与过程化设计和结构化程序设计密切相关.
A. 黑盒
B. **明盒**
C. 状态盒
D. 白盒

( ) 28-8. In cleanroom software engineering the structured programming approach is used to 400

A. refine data design
B. refine function design
C. refine usage test cases
D. both a and b

在cleanroom软件工程中, 结构化程序设计方法用于细化数据设计和功能设计, 帮助开发人员将系统的数据结构和功能分解为更小、更易管理和验证的单元, 提高软件质量和可靠性.

(D) 8. 在洁净室软件工程中, 结构化程序设计方法用于
A. 细化数据设计
B. 细化功能设计
C. 细化使用测试用例
D. **a 和 b 都是**

( ) 28-9. By using only structured programming constructs as you create a procedural design, you make the work of proving design correctness much easier. 401

在过程化设计中只使用结构化程序设计的构造(如顺序、选择、循环), 可以让证明设计正确性更容易. 结构化程序设计通过限制程序控制结构, 避免了goto等不规范跳转, 使程序逻辑清晰、层次分明. 这样每个模块的输入输出和行为都更容易理解和推理, 因此用数学方法证明程序满足需求时会更简单可靠.

(T) 9. 通过在过程化设计中仅使用结构化程序设计构造, 可以使证明设计正确性的工作变得更加容易.

( ) 28-10. Which of the following is not an advantage of using rigorous correctness verification of each refinement of the clear box design? 402

A. improves performance of code
B. produces better code than unit testing
C. reduces verification effort
D. results in near zero defect levels

严谨的正确性验证主要关注代码功能的正确性和无缺陷，不负责提升性能．性能优化需要专门的分析和改进方法，而不是仅靠正确性验证．选项B(比单元测试产生更好的代码)、C(减少验证工作量)、D(结果接近零缺陷)都是正确性验证带来的好处．

(A) 10. 下列哪一项不是对每一步清晰盒设计进行严格正确性验证的优点?
A. **提高代码性能**
B. 比单元测试产生更好的代码
C. 减少验证工作量
D. 实现接近零缺陷水平

( ) 28-11. Statistical use testing relies on probability distributions based on 403

A. mixture of control structures used in the program
B. order in which the module execute
C. the way software will actually be used
D. user interface design standards

统计使用测试(Statistical use testing)基于软件实际使用时各种操作发生的概率分布来设计测试用例，因此依赖于“软件实际被使用的方式”．它关注用户在真实环境下如何操作软件，通过模拟实际使用情况发现潜在问题．其他选项不涉及概率分布和实际使用，所以不正确．

(C) 11. 统计使用测试依赖于基于以下内容的概率分布
A. 程序中使用的控制结构的混合
B. 模块执行的顺序
C. **软件实际被使用的方式**
D. 用户界面设计标准

( ) 28-12. Certification of an increment is complete once it has passed the formal verification process. 404

通过形式化验证只是认证过程的一部分，完整的认证还包括实际测试、评审、满足用户需求等．因此，仅通过形式化验证不代表认证过程已经结束．

(F) 12. 一个增量的认证在通过正式验证过程后就完成了.

( ) 28-13. Which of the following models is part of the cleanroom certification process? 405

A. component model
B. sampling model
C. both a and b
D. none of the above

Cleanroom方法是一种强调软件高可靠性和缺陷预防的开发过程．在Cleanroom认证中，常用模型包括component model(组件模型)和sampling model(抽样模型)．component model用于描述各组件的可靠性，sampling model通过对软件运行进行抽样，估算整体可靠性．两者都是Cleanroom认证过程的一部分．

(C) 13. 以下哪种模型是无尘室认证过程的一部分?
A. 构件模型
B. 抽样模型
C. **a 和 b 都是**
D. 以上都不是

( ) 28-14. A data invariant is a set of conditions that are true during the execution of any function. 406

数据不变量(data invariant)是在特定时刻(如对象构造后、方法调用前后)必须为真的条件，不是在整个函数执行过程中都为真．题目说法不准确．

(F) 14. 数据不变式是在任何函数执行期间都为真的一组条件.

( ) 28-15. In some formal languages, stored data that the system accesses and alters is called a(n) 407

A. attribute
B. data structure
C. state
D. variant

在一些形式语言或系统中，系统访问和更改的存储数据被称为“状态(state)”．状态描述了系统在某一时刻的具体情况，所有相关数据的集合表示当前的状态，系统的运行通常是在不同状态之间转换．A. attribute(属性)指对象的某个特征，不是整体数据状态．B. data structure(数据结构)是组织和存储数据的方式，但不是特指系统当前数据的集合．D. variant(变体)通常指某种变化形式，也不是专指存储数据的集合.

(C) 15. 在某些形式化语言中，系统访问和更改的存储数据被称为
A. 属性
B. 数据结构
C. **状态**
D. 变量

( ) 28-16. In formal methods work, an action that reads or writes data to a state is called a(n) 408

A. actor
B. event
C. invariant
D. operation

在形式化方法中，action(动作)指对系统状态的数据进行读取或写入的行为，通常称为“操作”(operation)，它描述了系统状态如何被改变或查询．A“actor”是执行动作的实体，B“event”是系统中的事件，C“invariant”是不变量，都不符合题意.

(D) 16. 在形式化方法工作中，读取或写入状态数据的动作被称为
A. 参与者
B. 事件
C. 不变式
D. **操作**

( ) 28-17. What defines the circumstances in which a particular operation is valid? 409

A. data invariant
B. precondition
C. postcondition
D. state

前置条件(precondition)是指在执行某个操作前必须满足的条件或状态，只有这些条件满足时操作才有效．数据不变量(data invariant)是指数据在整个生命周期中应保持不变的属性．后置条件(postcondition)描述操作完成后应满足的条件．状态(state)表示对象或系统在某一时刻的情况.

(B) 17. 什么定义了特定操作有效的条件?
A. 数据不变式
B. **前置条件**
C. 后置条件
D. 状态

( ) 28-18. Using formal methods eliminates the need to write natural language commentary in the specification document. 410

即使采用形式化方法(formal methods)规范软件需求或设计，仍需要自然语言注释或说明．形式化方法虽然精确，但不一定被所有参与者(如客户、管理人员)理解，自然语言注释有助于解释和补充，使更多相关人员能正确理解规范内容，因此不能完全取代自然语言说明.

(F) 18. 使用形式化方法可以消除在规格说明文档中编写自然语言注释的需要.

( ) 29-1. Which of these are valid software configuration items? 411

A. case tools
B. documentation
C. executable programs
D. test data

软件配置项(Software Configuration Items, SCI)是指在软件开发中需要管理和控制的对象，包括源代码、文档、可执行程序、测试数据等．CASE工具(A)、文档(B)、可执行程序(C)、测试数据(D)在实际项目中都可以作为配置项进行管理，所以四项都是有效的软件配置项.

(ABCD) 1. 以下哪些是有效的软件配置项?
A. **CASE工具**
B. **文档**
C. **可执行程序**
D. **测试数据**

( ) 29-2. Which of the following is not considered one of the four important elements that should exist when a configuration management system is developed? 412

A. component elements
B. human elements
C. process elements
D. validation elements

配置管理系统的四个重要要素包括组成要素(component elements)、人员要素(human elements)、过程要素(process elements)以及相关工具和环境．A、B、C分别对应组件、人员和过程，都是配置管理的核心内容．D项“验证要素”主要与软件测试和质量保证相关，不属于配置管理系统开发的四大要素.

(D) 2. 下列哪一项不被认为是在开发配置管理系统时应存在的四个重要要素之一?
A. 组件要素
B. 人员要素
C. 过程要素
D. **验证要素**

( ) 29-3. Once a software engineering work product becomes a baseline it cannot be changed again. 413

基线(baseline)是经过正式评审和批准的工作产品版本，代表一个重要且受控的参考点，但不是不能再更改．如果需要修改(如修复缺陷或添加新需求)，可通过正式的变更控制流程进行更改，并形成新的基线．基线可以被更改，但要经过严格的变更管理.

(F) 3. 一旦软件工程工作产品成为基线，它就不能再被更改.

( ) 29-4. Which configuration objects would not typically be found in the project database? 414

A. design specification
B. marketing data
C. organizational structure description
D. test plans

项目数据库(或配置库)一般只包含与软件开发直接相关的技术文档，比如设计规格(A)和测试计划(D)．市场数据(B)和组织结构描述(C)属于管理或业务信息，不直接影响软件配置管理，所以不会放入项目数据库.

(BC) 4. 哪些配置对象通常不会出现在项目数据库中?
A. 设计规范
B. **市场数据**
C. **组织结构描述**
D. 测试计划

( ) 29-5. Modern software engineering practices usually attempt to maintain SCI's in a project database or repository. 415

现代软件工程实践通常会把SCI(软件配置项, Software Configuration Items)保存在项目数据库或代码仓库中．这样便于管理、跟踪和控制项目中的各种配置项，如源代码、文档、设计文件等，保证项目开发有序和版本可控．这也是配置管理的重要内容.

(T) 5. 现代软件工程实践通常尝试在项目数据库或代码库中维护SCI.

( ) 29-6. A data repository meta model is used to determine how 416

A. information is stored in the repository
B. well data integrity can be maintained
C. easily the existing model can be extended
D. all of the above

数据仓库元模型描述了数据在仓库中的组织、存储结构和关系，因此它决定了信息的存储方式(A)、影响数据完整性的维护(B)、以及模型在需要时的扩展性(C).

(D) 6. 数据仓库元模型用于确定如何
A. 信息在仓库中如何存储
B. 数据完整性能够被多好地维护
C. 现有模型能够多容易地扩展
D. **以上所有**

( ) 29-7. Many data repository requirements are the same as those for a typical database application. 417

许多数据仓库(data repository)的需求和典型数据库应用的需求相同. 数据仓库和普通数据库应用都关注数据的存储、检索、安全、完整性、备份和恢复等基本要求. 在需求层面，它们都需要支持数据一致性、并发控制、访问权限管理等功能，所以它们有很多相似的需求.

(T) 7. 许多数据仓库的需求与典型数据库应用的需求是相同的.

( ) 29-8. The ability to track relationships and changes to configuration objects is one of the most important features of the SCM repository. 418

SCM(软件配置管理)库的核心作用之一是跟踪配置项(如代码、文档、需求等)之间的关系，以及这些配置项在不同版本、不同时间点的变化. 通过跟踪，团队能清楚了解每个配置项的演变过程，方便溯源、恢复历史版本和管理依赖关系，这确实是SCM库非常重要的功能.

(T) 8. 跟踪配置对象之间的关系和变更的能力是SCM(软件配置管理)库最重要的特性之一.

( ) 29-9. Which of the following tasks is not part of software configuration management? 419

A. change control
B. reporting
C. statistical quality control
D. version control

软件配置管理(SCM)包括变更控制、报告、版本控制等，主要用于管理和记录软件的变更. 统计质量控制属于质量管理，不属于配置管理的任务.

(C) 9. 以下哪项任务不属于软件配置管理的内容?
A. 变更控制
B. 报告
C. **统计质量控制**
D. 版本控制

( ) 29-10. A basic configuration object is a _____ created by a software engineer during some phase of the software development process. 420

A. program data structure
B. hardware driver
C. unit of information
D. all of the above

题目考查配置项(configuration item)的基本概念. 基本配置对象是构成软件配置管理的最小信息单位，如文档、代码文件、规格说明书等. 这些是被单独标识和管理的信息单元，不仅仅指程序数据结构(A)或硬件驱动(B)，而是指所有单独管理的信息单位，所以选C.

(C) 10. 基本配置对象是软件工程师在软件开发过程的某个阶段创建的_____.
A. 程序数据结构
B. 硬件驱动程序
C. **信息单元**
D. 以上所有

( ) 29-11. Version control systems establish a change set as part of their primary functionality. 421

版本控制系统的主要功能是跟踪和管理文件的历史版本和变更，而"变更集"(change set)只是记录变更的一种方式，不是所有版本控制系统都必须用"变更集"这个概念. 将"建立变更集"作为主要功能是不准确的，主要功能应是管理和控制版本历史.

(F) 11. 版本控制系统将变更集作为其主要功能的一部分.

( ) 29-12. Change control is not necessary if a development group is making use of an automated project database tool. 422

即使开发团队用了自动化的项目数据库工具，变更控制仍然是必要的. 自动化工具能帮助跟踪和管理项目信息，但不能替代变更控制流程. 变更控制的主要作用是确保所有变更都经过评审、批准和记录，防止混乱和错误，提高软件质量和可维护性. 因此，变更控制是软件工程的基本要求，无论是否使用工具都不能省略.

(F) 12. 如果开发团队正在使用自动化项目数据库工具，则不需要变更控制.

( ) 29-13. When software configuration management is a formal activity the software configuration audit is conducted by the 423

A. development team
B. quality assurance group
C. senior managers
D. testing specialists

软件配置管理成为正式活动时，配置审核(software configuration audit)通常由质量保证组(quality assurance group)执行. 质量保证组负责独立检查和核实软件产品及其配置项是否符合标准和要求，确保完整性和一致性. 开发团队和测试专家主要负责开发和测试，独立性不足，管理层不直接参与审核工作.

(B) 13. 当软件配置管理成为一项正式活动时，软件配置审计由谁进行?
A. 开发团队
B. **质量保证小组**
C. 高级管理人员
D. 测试专家

( ) 29-14. The primary purpose of configuration status reporting is to 424

A. allow revision of project schedule and cost estimates by project managers
B. evaluate the performance of software developers and organizations
C. make sure that change information is communicated to all affected parties
D. none of the above

配置状态报告(configuration status reporting)主要是确保所有受影响的相关人员能够及时获得最新的变更信息. 它属于配置管理的重要环节，用于跟踪和传递软件项目中各配置项的变更状态，防止因信息滞后影响项目进展. 其他选项不涉及配置状态报告的核心目的.

(C) 14. 配置状态报告的主要目的是
A. 允许项目经理修订项目进度和成本估算
B. 评估软件开发人员和组织的绩效
C. **确保变更信息传达给所有受影响方**
D. 以上都不是

( ) 29-15. Configuration issues that need to be considered when developing Web and Mobile Apps include: 425

A. content
B. cost
C. people
D. politics

内容(content)是指应用中展示和处理的信息；成本(cost)涉及开发、部署和维护所需的资源；人员(people)包括开发、测试和运维等相关人员配置，这些都是开发Web和移动应用时必须考虑的配置因素. 政治(politics)有时会影响项目，但通常不属于技术配置问题，因此不选D.

(ABC) 15. 开发Web和移动应用时需要考虑的配置问题包括:
A. **内容**
B. **成本**
C. **人员**
D. 政策

( ) 29-16. Web and Mobile App configuration objects can be managed in much the same way as conventional software configuration objects except for: 426

A. content items
B. functional items
C. graphic items
D. user items

Web和移动应用的配置对象大多和传统软件类似，如功能项和图形项. 但Web和移动应用有特殊的配置对象——内容项(content items)，如网页文本、图片、音频等，这些内容变化快、更新频繁，管理方法不同于传统软件的配置对象.

(A) 16. Web 和移动应用的配置对象可以以与传统软件配置对象几乎相同的方式进行管理，除了:
A. **内容项**
B. 功能项
C. 图形项
D. 用户项

( ) 29-17. Content management establishes a process by which Web content is rendered on the user's display screen. 427

内容管理(Content management)指内容的创建、组织、存储和维护，不负责将内容在用户屏幕上显示. 内容的呈现(rendering)由前端技术(如HTML、CSS、JavaScript)和浏览器完成，不属于内容管理的范畴. 因此，内容管理不是内容渲染.

(F) 17. 内容管理建立了一个过程，通过该过程网页内容被呈现在用户的显示屏上.

( ) 29-18. Change management for Web and Mobile Apps is best handled in agile manner. 428

Web和移动应用的变更管理最好采用敏捷方式，因为这类应用需求变化快、更新频繁，用户反馈周期短. 敏捷方法强调对变化的快速响应和持续迭代，适合应对这类软件在市场和用户需求上的快速变化.

(T) 18. Web和移动应用的变更管理最好以敏捷方式进行.

( ) 29-19. One reason that version control is difficult for WebApps is that in an uncontrolled environment, you can have multiple authors making changes to the same files from multiple locations without any realizing it. 429

Web应用在版本控制时会遇到困难，因为在缺乏控制的环境下，多个作者可能会在不同地点对同一文件进行更改，彼此并不知情. Web应用开发常常需要多人协作，如果没有有效的版本控制工具和流程，就容易出现文件冲突和更改丢失的问题.

(T) 19. 版本控制对于Web应用来说较为困难的一个原因是，在一个不受控的环境中，可能有多个作者在不同地点对同一文件进行更改，而彼此并不知情.

( ) 29-20. Requiring developers to check Web configuration items in and out and sending affected stakeholders e-mail messages automatically are good

ways to deal with configuration auditing and reporting for WebApps. 430

让开发人员在管理Web配置项时使用签入签出机制，并自动向相关人员发送邮件通知，可以确保配置项的变更有记录，避免多人同时修改导致混乱．自动邮件通知让所有相关人员及时了解变更，增加透明度．这两者有助于配置的审计和追踪，方便后续报告和问题追溯，是Web应用配置管理中的好方法．

(T) 20. 要求开发人员签入和签出Web配置项，并自动向受影响的相关方发送电子邮件，是处理Web应用配置审计和报告的好方法．

(　) 30-1. Most technical software metrics described in this chapter represent indirect measures software attributes that are useful in the quantitative assessment of software quality. 431

本章介绍的大多数技术性软件度量是对软件属性的间接度量，这些度量用于定量评估软件质量．很多软件度量(如代码行数、圈复杂度、缺陷密度等)不能直接反映软件质量，而是通过这些数据间接反映如可维护性、可靠性等质量特性．

(T) 1. 本章中描述的大多数技术性软件度量都是对软件属性的间接度量，这些度量对于软件质量的定量评估是有用的．

(　) 30-2. Which these are reasons for using technical product measures during software development? 432

A. large body of scientific evidence supports their use
B. provides software engineers with an objective mechanism for assessing software quality
C. they allow all quality software quality information to be expressed unambiguously as a single number
D. all of the above

技术性产品度量可以为软件工程师提供客观机制评估软件质量．A选项不正确，目前没有大量科学证据绝对支持其使用；C选项也不正确，软件质量信息通常无法用单一数字完全表达．

(B) 2. 在软件开发过程中，使用技术产品度量的原因有哪些?
A. 有大量科学证据支持其使用
B. **为软件工程师提供了一个客观的机制来评估软件质量**
C. 它们允许所有软件质量信息都可以明确地用一个单一的数字表示
D. 以上全部

(　) 30-3. Which measurement activity is missing from the list below? 433

A. design
B. feedback
C. measurement
D. quantification

软件工程中的度量活动流程包括：度量(measurement)、量化(quantification)、反馈(feedback)、设计(design)．反馈是指将度量结果返回给相关人员，帮助改进后续开发和管理．选项中遗漏了反馈，这是度量流程中重要的一环．

(B) 3. 下列列表中缺少了哪项度量活动?
A. 设计
B. **反馈**
C. 度量
D. 量化

(　) 30-4. The Goal/Question/Metric (GQM) paradigm was developed as a technique for assigning blame for software failures. 434

GQM(目标/问题/度量)范式用于为软件开发设定目标，通过提出相关问题，然后确定合适的度量方法收集数据，以改进软件过程和产品质量．它帮助组织科学地度量和管理软件工程活动，不用于追查或归咎于个人的软件失败．

(F) 4. 目标/问题/度量(GQM)范式是为软件失败归责而开发的一种技术．

(　) 30-5. One of the most important attributes for a software product metric is that it should be 435

A. easy to compute
B. qualitative in nature
C. reliable over time
D. widely applicable

本题考查软件产品度量标准的重要属性．度量标准应具备多种特性，但“易于计算”(easy to compute)是最关键的．如果一个度量指标计算复杂、耗时或难以获取数据，开发人员和管理者难以在实际项目中经常使用它，会降低其实用性．简洁、易算的指标更可能被广泛采纳，从而更好地度量和改进软件质量．

(A) 5. 软件产品度量最重要的属性之一是它应该
A. **易于计算**
B. 具有定性特征
C. 随时间可靠
D. 广泛适用

(　) 30-6. In many cases metrics for one model may be used in later software engineering activities (e.g. design metrics may be used in test planning). 436

软件工程中某一阶段(如设计阶段)得到的度量指标可以在后续阶段(如测试规划)继续使用．例如，设计复杂度高的模块在测试时可能需要重点关注，因此设计阶段的度量结果能帮助测试阶段制定更有效的测试计划．这体现了度量贯穿软件开发生命周期、相互关联的特点．

(T) 6. 在许多情况下，一个模型的度量可以在后续的软件工程活动中使用(例如，设计度量可以用于测试计划)．

(　) 30-7. The function point metric is an example of metric that can be used to assist with technical decision-making based on the analysis model information, without making use of historical project data. 437

功能点(function point)度量在实际使用时，通常需要依赖历史项目数据来进行技术决策，如估算工作量、进度和成本．需要结合历史项目的数据，才能将功能点数转换为具体的资源需求或生产率等决策信息，不能只依赖分析模型的信息而不考虑历史数据．

(F) 7. 功能点度量是一种可以基于分析模型信息协助技术决策的度量方法，而无需使用历史项目数据．

(　) 30-8. The specification metrics proposed by Davis address which two characteristics of the software requirements? 438

A. functionality and performance
B. performance and completeness
C. specificity and completeness
D. specificity and functionality

Davis提出的需求规格度量主要衡量需求规格说明的“具体性”(specificity)和“完整性”(completeness)．具体性表示需求描述是否明确、无歧义，完整性指需求是否覆盖所有应有的功能和约束．

(C) 8. Davis提出的规格度量关注软件需求的哪两个特性?
A. 功能性和性能
B. 性能和完整性
C. **具体性和完整性**
D. 具体性和功能性

(　) 30-9. Architectural design metrics focus on 439

A. architectural structure
B. data structural relationships
C. internal module complexity
D. module effectiveness

Architectural design metrics主要关注软件系统的整体结构(architectural structure)和各模块的有效性(module effectiveness)，评估系统架构的合理性及模块实现需求的能力．B(数据结构关系)和C(内部模块复杂性)属于详细设计和编码层面，不是架构度量的主要内容．

(AD) 9. 架构设计度量关注于
A. **架构结构**
B. 数据结构关系
C. 模块内部复杂性
D. **模块有效性**

(　) 30-10. Which of the following is not a measurable characteristic of an object-oriented design? 440

A. completeness
B. efficiency
C. size
D. volatility

A. 完备性(completeness)、C. 大小(size)、D. 易变性(volatility)都是可以用具体指标度量的，如完备性用需求覆盖率，大小用类和方法数量，易变性用模块变化频率．B. 效率(efficiency)主要关注系统运行时的性能，通常在系统实现和运行阶段评估，不属于面向对象设计阶段的可度量属性．

(B) 10. 下列哪一项不是面向对象设计中可度量的特性?
A. 完整性
B. **效率**
C. 规模
D. 易变性

(　) 30-11. The depth of inheritance tree (DIT) metric can give an OO software designer a reading on the 441

A. attributes required for each class
B. completion time required for system implementation
C. complexity of the class hierarchy
D. level of object reusability achieved

DIT(继承树深度)表示从继承树根节点到某个类的路径长度，反映类层次结构的复杂度．DIT越大，类层次越复杂．DIT能让设计者了解类继承结构的复杂程度．B(系统实现完成时间)和DIT没有直接关系．

(B) 11. 继承树深度(DIT)度量可以为面向对象软件设计师提供关于以下哪方面的信息?
A. 每个类所需的属性
B. **系统实现所需的完成时间**
C. 类层次结构的复杂性
D. 实现的对象可重用性水平

(　) 30-12. Because the class is the dominant unit in OO systems there is no call for the definition of class-oriented metrics. 442

类是面向对象系统(OO systems)中最重要的构建单元，因此需要专门针对类的度量指标，如类的大小、复杂度、耦合度、内聚性等．这些度量有助于分析和改进系统设计．

(F) 12. 由于类是面向对象系统中的主要单元，因此没有必要定义面向类的度量．

(　) 30-13. If you encounter a class with a large responsibility (large class size or CS value) you should consider 443

A. making it a base class
B. making it a subclass
C. partitioning the class
D. starting a new class hierarchy

当一个类承担了过多的责任(如类的规模很大或复杂度高)，会导致代码难以维护、扩展和理解．应该将这个类拆分(partitioning the class)，即把大类的不同职责分离出来，形成多个更小、职责单一的类．这样每个类只关注自己的部分，符合单一职责原则，提高系统的可维护性和可扩展性．

(C) 13. 如果你遇到一个具有大量职责的类(类的规模大或CS值高)，你应该考虑
A. 将其作为基类
B. 将其作为子类
C. **对该类进行拆分**
D. 开始一个新的类层次结构

( ) 30-14. Component-level metrics include measures of 444

A. complexity
B. coupling
C. module cohesion
D. performance

组件级度量用于评估单个模块或组件的质量特性. 复杂度(complexity)、耦合(coupling)和内聚(module cohesion)是分析模块内部结构和与其他模块关系的重要指标. 性能(performance)通常是系统级或运行时度量，不属于组件级度量.

(ABC) 14. 构件级度量包括对以下方面的度量
A. **复杂度**
B. **耦合度**
C. **模块内聚性**
D. 性能

( ) 30-15. Because the class is the dominant unit in OO systems very few metrics have been proposed for operations that reside within a class. 445

大多数面向对象的软件度量主要关注类本身，如类的复杂度、耦合、内聚等. 针对类内部单个操作或方法的度量指标相对较少，所以说为类内部操作提出的度量很少这一说法是成立的.

(T) 15. 由于类是面向对象系统中的主要单元，因此很少有针对类内操作的度量被提出.

( ) 30-16. Interface metrics are use to assess the complexity of the module's input and output relationships with external devices. 446

接口度量(interface metrics)主要用于评估模块之间输入输出关系的复杂性，不是用于评估模块与外部设备关系. 接口度量关注软件内部模块通过接口传递数据的复杂程度，有助于分析模块间的耦合性和系统可维护性. 与外部设备的交互通常属于系统集成或硬件接口范畴，不是接口度量的直接对象.

(F) 16. 接口度量用于评估模块与外部设备之间输入和输出关系的复杂性.

( ) 30-17. Most WebApps can be easily characterized by judicious use of widely recognized suites of software metrics? 447

Web应用多样且复杂，传统软件度量方法(如代码行数、复杂度等)无法全面反映Web应用的特性，比如用户体验、响应速度、内容更新频率等. 因此，仅用常规度量工具难以准确、全面地描述Web应用.

(F) 17. 大多数Web应用程序可以通过明智地使用广泛认可的软件度量套件轻松表征吗?

( ) 30-18. Halstead's source code metrics are based on the number of 448

A. modules in the program
B. operands in the program
C. operators in the program
D. volume elements in the program

Halstead度量法用于衡量源代码复杂度，主要基于程序中的操作数(operands)和操作符(operators)数量. 操作数指变量、常量等，操作符指加减乘除、赋值等符号. 因此，Halstead度量主要依赖于操作数和操作符的数量，选项A和D不是其直接基础.

(BC) 18. Halstead 的源代码度量是基于以下数量的
A. 程序中的模块数
B. **程序中的操作数**
C. **程序中的运算符**
D. 程序中的体积元素

( ) 30-19. Software testing metrics fall into two broad categories 449

A. metrics that focus on defect removal effectiveness
B. metrics that focus on test coverage
C. metrics that estimate the duration of the testing process
D. metrics that predict the number of test cases required

软件测试度量主要有两类: 一类关注测试覆盖率(B)，用于衡量测试对代码或功能的覆盖程度; 另一类用于预测所需测试用例数量(D)，便于测试计划和资源分配. A和C虽然与测试相关，但不属于主要的两大类别.

(BD) 19. 软件测试度量大致分为两大类
A. 关注缺陷移除有效性的度量
B. **关注测试覆盖率的度量**
C. 估算测试过程持续时间的度量
D. **预测所需测试用例数量的度量**

( ) 30-20. The IEEE software maturity index (SMI) is used to provide a measure of the 450

A. maintainability of a software product based on its availability
B. relative age of a software product being considered for retirement
C. reliability of a software product following regression testing
D. stability of a software product as it is modified during maintenance

IEEE软件成熟度指数(SMI)用于衡量软件在维护过程中多次修改后的功能稳定性. SMI值越高，表示经过修改、修复和升级后, 软件的变化减少, 功能更加成熟和稳定. SMI关注的是软件在持续修改中"稳定性"的度量，不直接衡量可维护性、相对年龄或回归测试后的可靠性.

(D) 20. IEEE软件成熟度指数(SMI)用于衡量
A. 基于可用性的软件产品可维护性
B. 正在考虑退役的软件产品的相对年龄
C. 回归测试后软件产品的可靠性
D. **软件产品在维护过程中被修改时的稳定性**

( ) 31-1. Effective software project management focuses on 451

A. people, performance, payoff, product
B. people, product, performance, process
C. people, product, process, project
D. people, process, payoff, product

有效的软件项目管理主要关注人员(people)、产品(product)、过程(process)和项目(project)四个方面. 人员指项目团队及其沟通协作，产品是开发的目标软件，过程是开发和管理的方法与步骤，项目则是对整个开发活动的管理. 选项C全面准确地涵盖了这四个关键要素.

(C) 1. 有效的软件项目管理关注于
A. 人员、绩效、回报、产品
B. 人员、产品、绩效、过程
C. **人员、产品、过程、项目**
D. 人员、过程、回报、产品

( ) 31-2. Organizations that achieve high levels of maturity in people management have a higher likelihood of implementing effective software engineering processes. 452

题目强调了人在软件开发过程中的重要性. 组织在人力资源管理上有较高成熟度，如完善的培训、激励、团队协作机制，会提升员工的积极性和能力，有助于更好地执行和改进软件工程流程. 高水平的人才管理通常能带来更有效的软件工程过程.

(T) 2. 在人员管理方面达到高成熟度的组织，更有可能实施有效的软件工程过程.

( ) 31-3. The first step in project planning is to 453

A. determine the budget.
B. select a team organizational model.
C. determine the project constraints.
D. establish the objectives and scope.

项目规划的第一步是明确项目的目标和范围，即要清楚项目要实现什么、目的是什么、包括哪些内容、不包括哪些内容. 在确定目标和范围后，才能进一步制定预算、团队结构和约束条件. 确定预算、选择团队结构、确定约束条件等，都是在明确目标和范围后进行的.

(D) 3. 项目规划的第一步是
A. 确定预算.
B. 选择团队组织模型.
C. 确定项目约束条件.
D. **确立目标和范围.**

( ) 31-4. Process framework activities are populated with 454

A. milestones
B. work products
C. QA points
D. all of the above

过程框架活动包括项目进展的重要节点(milestones)、活动过程中产生的工作成果(work products)，以及确保质量的检查点(QA points). 这些元素共同填充和定义过程框架活动.

(D) 4. 流程框架活动由以下内容填充
A. 里程碑
B. 工作产品
C. 质量保证点
D. **以上全部**

( ) 31-5. Project management is less important for modern software development since most projects are successful and completed on time. 455

项目管理在现代软件开发中依然非常重要，因为很多软件项目仍会遇到延期、超支或未达预期目标等问题. 良好的项目管理有助于规划进度、分配资源、控制风险，提高项目成功率，所以不能因为部分项目成功就忽视项目管理的重要性.

(F) 5. 由于大多数项目都成功并按时完成，项目管理对现代软件开发来说不那么重要.

( ) 31-6. Which of the following is not considered a stakeholder in the software process? 456

A. customers
B. end-users
C. project managers
D. sales people

stakeholder(利益相关者)是指对软件项目有直接利益或影响的人或组织. 常见利益相关者有客户(customers)、最终用户(end-users)、项目经理(project managers)，他们直接参与或受软件过程影响. 销售人员(sales people)通常不直接参与软件开发过程，也不直接受其影响，因此一般不被视为该过程的利益相关者.

(D) 6. 以下哪一项不被认为是软件过程中的相关方?
A. 客户
B. 最终用户
C. 项目经理
D. **销售人员**

( ) 31-7. The best person to hire as a project team leader is the most competent software engineering practitioner available. 457

虽然技术能力重要，但项目团队领导还需要具备领导、沟通、协调和管理能力. 最优秀的技术人员不一定擅长管理和领导团队，因此不能仅以技术水平作为团队领导的唯一标准.

(F) 7. 作为项目团队负责人，最适合聘用的人是可用的最有能力的软件工程从业者.

( ) 31-8. The best project team organizational model to use when tackling extremely complex problems is the 458

A. closed paradigm
B. open paradigm
C. random paradigm
D. synchronous paradigm

本题考查项目团队的组织模型. open paradigm(开放范式)强调团队成员之间的广泛交流、协作和信息共享，有助于集思广益，灵活应对复杂多变的需求和挑战，适合处理极其复杂的问题. closed paradigm(封闭范式)适用于分工明确、任务较简单的项目; random paradigm(随机范式)和synchronous paradigm(同步范式)不适合高复杂度项目.

(B) 8. 在解决极其复杂问题时，最适合采用的项目团队组织模型是
A. 封闭范式
B. **开放范式**
C. 随机范式
D. 同步范式

( ) 31-9. Which factors should be considered in choosing the organizational structure for a software team? 459

A. degree of communication desired
B. predicted size of the resulting program
C. rigidity of the delivery date
D. size of the project budget

选择软件团队组织结构时，要考虑沟通需求(A)，因为沟通频率和方式影响组织结构的层次; 还要考虑预期软件规模(B)，项目越大，团队结构通常越复杂; 交付日期的严格性(C)也很重要，交付时间紧，团队结构需支持高效协作和快速响应. 这些因素都直接影响团队组织方式. 项目预算大小(D)主要影响资源投入，不直接决定组织结构.

(ABC) 9. 在选择软件团队的组织结构时，应考虑哪些因素?
A. **期望的沟通程度**
B. **预期生成程序的规模**
C. **交付日期的严格性**
D. 项目预算的规模

( ) 31-10. One of the best ways to avoid frustration during the software development process is to 460

A. give team members more control over process and technical decisions.
B. give team members less control over process and technical decisions.
C. hide bad news from the project team members until things improve.
D. reward programmers based on their productivity.

让团队成员有更多对流程和技术决策的控制权，可以提升主动性和积极性，减少因被动执行或决策不透明带来的挫败感. 这样成员会有更强的归属感和责任感，更好地解决问题，提高项目成功率. 其他选项可能降低士气，或导致信息不透明、不公平.

(A) 10. 在软件开发过程中，避免挫折感的最佳方法之一是
A. **给予团队成员更多对流程和技术决策的控制权.**
B. 给予团队成员更少对流程和技术决策的控制权.
C. 在情况好转之前，对项目团队成员隐瞒坏消息.
D. 根据程序员的生产力给予奖励.

( ) 31-11. Small agile teams have no place in modern software development. 461

小型敏捷团队在现代软件开发中非常重要. 敏捷方法强调小团队的高效协作、快速响应变化和持续交付价值，许多现代公司(如互联网公司、初创企业)都采用小型敏捷团队提升开发效率和产品质量. 所以说小型敏捷团队在现代软件开发中没有地位是错误的.

(F) 11. 小型敏捷团队在现代软件开发中没有立足之地.

( ) 31-12. Which of these software characteristics is not a factor contributing to project coordination difficulties? 462

A. interoperability
B. performance
C. scale
D. uncertainty

项目协调困难通常与信息交流、任务分配、技术兼容性、需求变化等有关. A. interoperability(互操作性)指不同系统之间能否协同工作，涉及多个团队和系统的协调，会导致项目协调困难; C. scale(规模)项目规模越大，人员、模块、需求越多，协调难度越大; D. uncertainty(不确定性)需求、技术或环境的不确定性会增加团队协调的难度; B. performance(性能)主要关注软件运行效率，不直接影响团队或项目管理的协调难度.

(B) 12. 下列哪些软件特性不是导致项目协调困难的因素?
A. 互操作性
B. **性能**
C. 规模
D. 不确定性

( ) 31-13. Which of these software characteristics are used to determine the scope of a software project? 463

A. context, lines of code, function
B. context, function, communication requirements
C. information objectives, function, performance
D. communications requirements, performance, information objectives

确定软件项目范围时，主要关注软件要实现的功能(function)、目标(information objectives)和性能要求(performance). 这些特性可以明确项目的边界和预期成果. context、代码行数、通信需求等不是直接用来界定项目范围的核心特性.

(C) 13. 以下哪些软件特性用于确定软件项目的范围?
A. 上下文、代码行数、功能
B. 上下文、功能、通信需求
C. **信息目标、功能、性能**
D. 通信需求、性能、信息目标

( ) 31-14. The major areas of problem decomposition during the project scoping activity are the 464

A. customer workflow
B. functionality to be delivered
C. process used to deliver functionality
D. software process model

项目范围界定(project scoping)活动中，问题分解的主要领域包括要交付的功能(B)，即软件需实现的功能，是范围界定的核心内容，以及交付功能所用的过程(C)，即开发这些功能所采用的具体过程，这也是范围界定时需明确的问题. A(客户工作流程)和D(软件过程模型)虽然相关，但不是在范围界定时直接需要分解的主要领域.

(BC) 14. 在项目范围界定活动中，问题分解的主要领域是
A. 客户工作流程
B. **要交付的功能**
C. **用于交付功能的过程**
D. 软件过程模型

( ) 31-15. Product and process decomposition occurs simultaneously as the project plan evolves. 465

在软件工程中，项目计划制定过程中，团队会同时进行产品分解(将产品分成各功能模块)和过程分解(将开发流程细化为具体任务和步骤). 这两个分解过程相互关联，产品结构影响过程安排，过程细化也可能影响产品实现方式，因此它们会随着项目计划的完善而同步演进.

(T) 15. 随着项目计划的推进，产品和过程的分解是同时进行的.

( ) 31-16. When can selected common process framework activities be omitted during process decomposition? 466

A. when the project is extremely small in size
B. any time the software is mission critical
C. rapid prototyping does not require their use
D. never the activities are invariant

软件过程框架活动是“invariant”，即不变的，不能省略. 不管项目大小、重要性或开发方式如何，过程框架中的基本活动(如通信、计划、建模、构建、部署)都必须执行，不能因为项目特殊情况而省略. 这些活动对软件工程过程来说是基础和必要的.

(D) 16. 在进行过程分解时，何时可以省略选定的通用过程框架活动?
A. 当项目规模极小时
B. 只要软件是关键任务型的
C. 快速原型开发不需要使用它们
D. **从不，这些活动是不可变的**

( ) 31-17. How does a software project manager need to act to minimize the risk of software failure? 467

A. double the project team size
B. request a large budget
C. start on the right foot
D. track progress

选项C“start on the right foot”指项目初期要做好规划，明确目标和流程，打下良好基础，有利于项目顺利进行. 选项D“track progress”是指在项目过程中持续跟踪进度，及时发现和解决问题，保证项目按计划推进，这两种做法都能有效降低失败风险. A、B并不能直接降低风险，反而可能增加管理难度或造成资源浪费.

(CD) 17. 软件项目经理应如何行动以最小化软件失败的风险?
A. 增加项目团队规模
B. 申请大量预算
C. **从一开始就做对**
D. **跟踪进度**

( ) 31-18. The W5HH principle contains which of the following questions? 468

A. Why is the system being developed?
B. What will be done by whom?
C. Where are they organizationally located?
D. How much of each resource is required?

W5HH原则帮助项目管理者全面考虑项目目标和过程，包含七个核心问题: Why(为什么做)、What(做什么)、When(什么时候做)、Who(谁负责)、Where(在什么地方做)、How(怎么做)、How much(需要多少资源). A对应Why，C对应Where，D对应How much，都是W5HH原则中的问题. B虽然涉及Who，但W5HH强调的是“谁负责”，而不是“由谁做什么”这种具体分工.

(ACD) 18. W5HH原则包含以下哪些问题?
A. **为什么要开发该系统?**
B. 谁将做什么?
C. **他们在组织中的位置在哪里?**
D. **每种资源需要多少?**

( ) 31-19. Which of these are critical practices for performance-based project management? 469

A. assessing product usability
B. defect tracking against quality targets
C. empirical cost estimation
D. formal risk management

B(针对质量目标进行缺陷跟踪)有助于监控和改进产品质量，C(经验成本估算)确保项目在预算内进行，D(正式的风险管理)帮助识别和应对项目中的潜在风险，这三项都是直接关系到项目绩效的管理活

动. A(评估产品可用性)虽然重要，但更偏向于产品本身，而不是项目执行过程的管理，因此不是关键实践.

(BCD) 19. 以下哪些是基于绩效的项目管理中的关键实践?
A. 评估产品可用性
B. **针对质量目标的缺陷跟踪**
C. **基于经验的成本估算**
D. **正式的风险管理**

( ) 32-1. Which of these are valid reasons for measuring software processes, products, and resources? 470

A. to characterize them
B. to evaluate them
C. to price them
D. to improve them

度量软件过程、产品和资源的主要目的是了解和描述(characterize)、评估(evaluate)和改进(improve)它们. 度量有助于发现问题、质量控制和流程优化. 定价(price)不是度量的直接目的，虽然定价可能会参考度量数据，但度量本身并不是为定价而设.

(ABD) 1. 以下哪些是对软件过程、产品和资源进行度量的有效理由?
A. **对其进行表征**
B. **对其进行评估**
C. 对其进行定价
D. **对其进行改进**

( ) 32-2. Process indicators enable a software project manager to 471

A. assess the status of an on-going project
B. track potential risks
C. adjust work flow or tasks
D. none of the above

过程指标(process indicators)用于衡量和分析软件开发过程的效率和效果，如进度和缺陷密度，帮助项目经理了解过程表现、发现问题并改进. 选项A、B、C都属于过程指标的实际作用：A是评估项目状态，B是追踪风险，C是调整流程或任务. 所以A、B、C都可以通过过程指标实现，答案D不正确.

(D) 2. 过程指标使软件项目经理能够
A. 评估正在进行项目的状态
B. 跟踪潜在风险
C. 调整工作流程或任务
D. **以上都不是**

( ) 32-3. Public metrics are used 472

A. to evaluate the performance of software development teams.
B. to appraise the performance of individual team members.
C. to make strategic changes to the software process.
D. to make tactical changes during a software project.

Public metrics 是团队或组织层面对外公开的软件开发过程和产品度量数据，主要用于帮助团队和管理层了解项目整体状况并做决策. 选项C和D正确，因为这些度量既可用于长期过程改进，也可在项目执行中及时发现和纠正问题. A和B侧重于个人或团队绩效评价，这不是public metrics的主要用途.

(CD) 3. 公共度量的用途
A. 用于评估软件开发团队的绩效.
B. 用于评价团队中个人成员的绩效.
C. **用于对软件过程进行战略性调整.**
D. **用于在软件项目中进行战术性调整.**

( ) 32-4. Which of the following items are not measured by software project metrics? 473

A. inputs
B. markets
C. outputs
D. results

软件项目度量(metrics)关注项目的输入(如人力、资源)、输出(如代码、文档)、以及结果(如产品质量、缺陷率)等内容. 市场(markets)属于外部环境因素，不是项目本身的度量对象，所以“markets”不属于软件项目度量衡量的内容.

(B) 4. 下列哪些项目不是通过软件项目度量来衡量的?
A. 输入
B. **市场**
C. 输出
D. 结果

( ) 32-5. Software quality and functionality must be measured indirectly. 474

软件质量和功能性无法像温度、重量等物理量一样用仪器直接测量. 它们通常通过用户体验、缺陷率、性能指标、满意度调查等多种间接方法进行评估，因此只能间接测量，而无法直接测量.

(T) 5. 软件质量和功能性必须通过间接方式进行度量.

( ) 32-6. Which of following are advantages of using LOC (lines of code) as a size-oriented metric? 475

A. LOC is easily computed.
B. LOC is a language dependent measure.
C. LOC is a language independent measure.
D. LOC can be computed before a design is completed.

LOC(代码行数)的优点是很容易计算，只需统计源代码行数即可. B是缺点，因为不同编程语言下，LOC值会有差异. C错误，LOC不是语言无关的. D错误，设计完成和实际编写代码后才能准确计算LOC.

(A) 6. 以下哪项是使用LOC(代码行数)作为面向规模度量的优点?
A. **LOC易于计算.**
B. LOC是一种依赖于编程语言的度量.
C. LOC是一种独立于编程语言的度量.
D. LOC可以在设计完成之前计算.

( ) 32-7. Which of the following are advantages of using function points (FP) as a measure of the functionality delivered by a software application? 476

A. FP is easily computed.
B. FP is a language dependent measure.
C. FP is a language independent measure.
D. FP can be computed before a design is completed.

Function Points(功能点)用于衡量软件功能规模. 优点包括：C. FP与编程语言无关(language independent)，即无论使用哪种语言，度量结果一致. D. FP可以在设计完成前，通过需求规格说明书进行估算，无需等到详细设计或编码阶段. A错误，FP计算需分析需求，不是特别容易. B错误，FP强调与语言无关.

(CD) 7. 下列哪些是使用功能点(FP)作为衡量软件应用交付功能的优点?
A. FP 易于计算.
B. FP 是一种依赖于编程语言的度量.
C. **FP 是一种独立于编程语言的度量.**
D. **FP 可以在设计完成之前计算.**

( ) 32-8. There is no need to reconcile LOC and FP measures since each in meaningful in its own right as a project measure. 477

LOC(代码行数)和FP(功能点)是两种不同的软件规模度量方法，分别从不同角度衡量项目. 在实际项目管理和估算时，常常需要将两者结合，对比和校验，以提升估算的准确性和一致性. 只使用其中一种可能导致度量出现偏差，因此有必要对两者进行协调和相互验证.

(F) 8. 没有必要协调LOC(代码行数)和FP(功能点)度量，因为每种度量作为项目度量本身都是有意义的.

( ) 32-9. Object-Oriented project measures may be combined with historical project data to provide metrics that aid in project estimation. 478

面向对象项目的度量(如类的数量、方法数量、继承层次深度等)可以与历史项目的实际数据结合分析，帮助更准确地估算新项目的工作量、进度和资源需求. 这是项目管理中的常用做法.

(T) 9. 面向对象项目的度量可以与历史项目数据结合，以提供有助于项目估算的度量指标.

( ) 32-10. Use-Case oriented metrics are computed directly from UML diagrams they are often used as normalization measures. 479

用例导向的度量虽然和UML图有关，但不能完全、直接从UML图中提取，还需要对需求的理解和解释. 它们通常作为软件规模或复杂度估算的基础，而不是标准的归一化度量.

(F) 10. 用例导向度量是直接从UML图中计算得出的，它们通常被用作归一化度量.

( ) 32-11. Which of the following is not a measure that can be collected from a Web application project? 480

A. Customization index
B. Number of dynamic objects
C. Number of internal page links
D. Number of static web pages

选项A“Customization index”(定制化指数)不是Web应用常用的度量指标，没有统一、标准的定义，也不常用于衡量项目特性. B(动态对象数量)、C(内部页面链接数量)、D(静态网页数量)都是Web项目中常见且易采集的度量指标，能反映网站结构和规模. 因此，A不是Web应用项目中常见或标准的可采集度量.

(A) 11. 下列哪一项不是可以从Web应用项目中收集的度量指标?
A. **定制化指数**
B. 动态对象数量
C. 内部页面链接数量
D. 静态网页数量

( ) 32-12. Which of the following software quality factors is most likely to be affected by radical changes to computing architectures? 481

A. operation
B. transition
C. revision
D. none of the above

软件质量因素(如operation、transition、revision)都会受到计算机体系结构根本性变化的影响. 体系结构的激烈变动会同时影响操作性、可移植性、可维护性，而不是某一个单独的质量因素，因此没有哪一个质量因素最容易受到影响.

(D) 12. 下列哪项软件质量因素最有可能受到计算机体系结构根本性变化的影响?
A. 运行性
B. 可迁移性
C. 可修改性
D. **以上都不是**

( ) 32-13. Which of the following provide useful measures of software quality? 482

A. correctness, performance, integrity, usability
B. reliability, maintainability, integrity, sales
C. correctness, maintainability, size, satisfaction
D. correctness, maintainability, integrity, usability

软件质量通常从多个维度衡量，包括正确性(correctness)、可维护性(maintainability)、完整性(integrity)、可用性(usability)等. 选项D包含了这四个核心指标，都是评价软件质量非常有用的标准. 其他选项中的performance(性能)、sales(销量)、size(规模)、satisfaction(满意度)虽然相关，但不是通用的质量度量标准.

(D) 13. 下列哪些提供了有用的软件质量度量?
A. 正确性、性能、完整性、可用性
B. 可靠性、可维护性、完整性、销售量
C. 正确性、可维护性、规模、满意度
D. **正确性、可维护性、完整性、可用性**

( ) 32-14. A software quality metric that can be used at both the process and project levels is defect removal efficiency (DRE). 483

缺陷清除效率(Defect Removal Efficiency, DRE)是一种既可用于过程级，也可用于项目级的软件质量度量. DRE衡量开发过程中发现并移除的缺陷比例，可用于评估具体项目的质量控制效果，也可分析整个开发过程的质量管理能力，因此适用于过程和项目两个层面.
(T) 14. 一个可以在过程和项目两个层面使用的软件质量度量是缺陷去除效率(DRE).

( ) 32-15. Why is it important to measure the process of software engineering and software it produces? 484

A. It is really not necessary unless the project is extremely complex.
B. To determine costs and allow a profit margin to be set.
C. To determine whether a software group is improving or not.
D. To make software engineering more like other engineering processes.

测量软件工程过程和产出的软件很重要，因为这样才能判断软件开发团队的工作是否在改进，比如效率和质量是否提升. 如果不进行度量，就无法发现问题、持续改进和优化过程. 度量的核心目的是促进持续改进.
(C) 15. 为什么要对软件工程过程及其产出的软件进行度量?
A. 除非项目极其复杂，否则其实没有必要.
B. 为了确定成本并设定利润空间.
C. **为了判断一个软件团队是否在改进.**
D. 为了让软件工程更像其他工程过程.

( ) 32-16. To be an effective aid in process improvement the baseline data used must be: 485

A. based on reasonable guestimates from past projects
B. measured consistently across projects
C. drawn from similar projects
D. based on all previously completed projects

基线数据应在不同项目中一致测量(B)，便于比较和分析；同时应来自与当前项目相似的项目(C)，这样数据才有参考价值. 仅凭猜测(A)或把所有项目混在一起(D)，数据的代表性和可靠性会降低.
(BC) 16. 为了有效地辅助过程改进，所使用的基线数据必须是:
A. 基于以往项目的合理估算
B. **在各项目之间一致地进行度量**
C. **来自相似的项目**
D. 基于所有已完成的项目

( ) 32-17. Baseline data must be collected in an ongoing manner and cannot be computed by formal study of historical project data. 486

基线数据可以通过持续收集获得，也可以通过分析和正式研究历史项目数据来计算. 题干说"不能通过对历史项目数据的正式研究来计算"是错误的.
(F) 17. 基线数据必须以持续的方式收集，不能通过对历史项目数据的正式研究来计算.

( ) 32-18. Small software organizations are not likely to see any economic return from establishing software metrics program. 487

小型软件组织建立软件度量(metrics)项目同样能获得经济回报. 无论组织规模，软件度量都能改进开发过程、提升质量、减少错误和返工，带来成本节约和经济效益. 小型组织也可以利用数据度量发现问题、优化资源分配、提高效率.
(F) 18. 小型软件组织不太可能通过建立软件度量程序获得任何经济回报.

( ) 32-19. The software metrics chosen by an organization are driven by the business or technical goals an organization wishes to accomplish. 488

软件度量(metrics)的选择应服务于组织的业务或技术目标. 例如，组织希望提高软件质量时，会关注缺陷密度、测试覆盖率等度量；关注开发效率时，可能关注生产率、交付周期等. 因此，度量指标应围绕组织目标确定，而不是随意选择.
(T) 19. 一个组织选择的软件度量是由其希望实现的业务或技术目标所驱动的.

( ) 33-1. Since project estimates are not completely reliable, they can be ignored once a software development project begins. 489

项目估算虽然不完全准确，但为资源分配、进度安排和风险管理提供重要参考. 在项目执行中，估算还需与实际进展对比，用于调整和决策. 如果忽略估算，项目会失去管理依据，容易导致进度延误和成本超支，因此估算不能被忽略.
(F) 1. 由于项目估算并不完全可靠，因此在软件开发项目开始后可以忽略它们.

( ) 33-2. The objective of software project planing is to 490

A. convince the customer that a project is feasible.
B. make use of historical project data.
C. enable a manager to make reasonable estimates of cost and schedule.
D. determine the probable profit margin prior to bidding on a project.

软件项目计划的主要目标是帮助项目经理合理估算项目的成本和进度，便于后续管理和控制. A、B只是计划的辅助作用，D属于企业经营层面，不是项目计划的核心目标.
(C) 2. 软件项目计划的目标是
A. 说服客户项目是可行的.
B. 利用历史项目数据.
C. **使管理者能够对成本和进度做出合理的估算.**
D. 在对项目进行投标前确定可能的利润率.

( ) 33-3. The project scope is defined as a means of bounding the system 491

A. functionality
B. performance
C. costs
D. schedule

项目范围(project scope)指确定软件系统边界，包括系统应具备的功能和性能要求. 功能性(A)描述系统应完成的任务，性能(B)规定系统应达到的速度、容量等. 成本(C)和进度(D)属于项目管理内容，不用于界定系统边界.
(AB) 3. 项目范围被定义为限定系统的手段
A. **功能性**
B. **性能**
C. 成本
D. 进度安排

( ) 33-4. Software feasibility is based on which of the following 492

A. business and marketing concerns
B. scope, constraints, market
C. technology, finance, time, resources
D. technical prowess of the developers

软件可行性分析关注项目在技术、财务、时间和资源等方面的可行性，即项目是否能在现有技术条件、预算、时间进度和资源支持下完成. 选项A和B只涉及部分相关因素，不够全面；选项D只强调开发者技术，不能全面评估可行性.
(C) 4. 软件可行性基于以下哪些方面
A. 业务和市场因素
B. 范围、约束、市场
C. **技术、财务、时间、资源**
D. 开发者的技术能力

( ) 33-5. The number of people required for a software project is determined 493

A. after an estimate of the development effort is made.
B. by the size of the project budget.
C. from an assessment of the technical complexity of the system.
D. all of the above

确定项目所需人数，首先要进行开发工作量的估算，明确整个项目需要多少人力工时，才能合理分配人员. 项目预算和技术复杂度也会对人员分配产生影响，但最直接的依据是工作量估算.
(A) 5. 软件项目所需人数的确定依据是
A. **在对开发工作量进行估算之后.**
B. 由项目预算的规模决定.
C. 通过对系统技术复杂性的评估得出.
D. 以上全部

( ) 33-6. Reusable software components must be 494

A. catalogued for easy reference.
B. standardized for easy application.
C. validated for easy integration.
D. all of the above

可复用软件构件(Reusable software components)应具备几个基本条件：首先，为了方便查找和引用，需要被编目(A)；其次，为了便于在不同环境中应用，需要有统一的标准(B)；最后，为确保能够顺利集成到其他系统中，还必须经过验证(C). 因此，A、B、C三项都是必不可少的.
(D) 6. 可复用的软件组件必须
A. 便于查找地编目.
B. 便于应用地标准化.
C. 便于集成地验证.
D. **以上全部**

( ) 33-7. The software engineering environment (SEE) consists of which of the following? 495

A. customers
B. developers
C. hardware platforms
D. software tools

SEE(软件工程环境)指的是支持软件开发和维护的各种硬件和软件资源. 选项C(硬件平台)和D(软件工具)属于SEE，因为它们提供开发、测试、运行等必要的支持环境. A(客户)和B(开发者)属于人员，不是环境的组成部分.
(CD) 7. 软件工程环境(SEE)包括以下哪些内容?
A. 客户
B. 开发人员
C. **硬件平台**
D. **软件工具**

( ) 33-8. Software project estimation techniques can be broadly classified under which of the following headings? 496

A. automated processes
B. decomposition techniques

C. empirical models
D. regression models

软件项目估算技术主要分为分解技术(B)和经验模型(C). 分解技术指将项目拆分为较小的部分分别估算再汇总, 经验模型是基于历史项目数据用统计和数学方法估算. A"自动化过程"和D"回归模型"不是主要分类, 回归模型只是经验模型的一种实现方式.

(BC) 8. 软件项目估算技术大致可以归类于以下哪些类别?
A. 自动化过程
B. **分解技术**
C. **经验模型**
D. 回归模型

( ) 33-9. The size estimate for a software product to be built must be based on a direct measure like LOC. 497

软件规模估算不仅可以用直接度量(如代码行数LOC), 还可以用间接度量, 如功能点(Function Points)、用例点等. 间接度量方法能更好地反映软件的功能和复杂度, 有时比代码行数更适合早期估算. 因此, 规模估算不一定要依赖代码行数这样的直接度量.

(F) 9. 对于要构建的软件产品的规模估算, 必须基于像代码行数(LOC)这样的直接度量.

( ) 33-10. Problem-based estimation is based on problem decomposition which focuses on 498

A. information domain values
B. project schedule
C. software functions
D. process activities

问题分解法(problem-based estimation)通过将软件开发问题分解为更小的部分来估算工作量和成本. 它关注"信息域值"(如输入、输出、文件等)和"软件功能"(如需要实现的功能), 这些体现了系统要解决的问题和目标. 项目进度(B)和过程活动(D)属于项目管理和开发过程, 不是问题分解法的关注点.

(AC) 10. 基于问题的估算是基于问题分解, 其重点在于
A. **信息域值**
B. 项目进度表
C. **软件功能**
D. 过程活动

( ) 33-11. LOC-based estimation techniques require problem decomposition based on 499

A. information domain values
B. project schedule
C. software functions
D. process activities

LOC(Lines of Code)估算方法需要将软件系统分解为不同的软件功能, 然后分别估算每个功能的代码行数, 最后汇总得到总LOC. 问题分解依据是"软件功能"(software functions). 信息域值、项目进度、过程活动都不是LOC估算分解的依据.

(C) 11. 基于代码行数(LOC)的估算技术需要基于什么进行问题分解?
A. 信息域值
B. 项目进度安排
C. **软件功能**
D. 过程活动

( ) 33-12. FP-based estimation techniques require problem decomposition based on 500

A. information domain values
B. project schedule
C. software functions
D. process activities

FP(Function Point, 功能点)估算技术要求将问题分解为一组信息域值, 如输入、输出、查询、内部文件和外部接口等, 这些都属于信息域内容. 通过统计这些信息域的数量和复杂度, 估算软件规模. FP估算是基于信息域值进行问题分解的.

(A) 12. 基于功能点(FP)的估算技术需要基于以下哪项进行问题分解
A. **信息域值**
B. 项目进度表
C. 软件功能
D. 过程活动

( ) 33-13. Process-based estimation techniques require problem decomposition based on 501

A. information domain values
B. project schedule
C. software functions
D. process activities

基于过程的估算技术(process-based estimation techniques)在估算工作量或成本时, 需要先将开发工作分解为更小部分. 选项C"software functions"(软件功能)和D"process activities"(过程活动)都是常用的分解方式. 可以把软件划分为功能模块, 或按照开发过程(如需求分析、设计、编码、测试等)分解, 以更准确地估算每一部分所需的资源和时间. A和B不是常见的问题分解依据.

(CD) 13. 基于过程的估算技术需要基于以下哪项进行问题分解
A. 信息域值
B. 项目进度表
C. **软件功能**
D. **过程活动**

( ) 33-14. Unlike a LOC or function point each person's "use-case" is exactly the same size. 502

用例(use-case)描述用户与系统的交互, 不同用例的复杂度、步骤和范围各不相同, 所以用例的大小(复杂度和工作量)不会都一样. 每个人编写的用例内容和细节也可能不同, 因此用例不会完全相同. 相比之下, 代码行数(LOC)和功能点是更客观的度量单位, 而用例不是固定大小的.

(F) 14. 与代码行数(LOC)或功能点不同, 每个人的"用例"大小完全相同.

( ) 33-15. When agreement between estimates is poor the cause may often be traced to inadequately defined project scope or inappropriate productivity data. 503

当估算结果差异较大时, 常见原因包括项目范围定义不清或生产率数据不合适. 如果项目范围没有明确规定, 各自对工作内容理解不同, 导致估算差异大; 使用与实际项目不符的生产率数据也会影响估算准确性. 因此, 要保证估算一致, 需明确项目范围并使用可靠的生产率数据.

(T) 15. 当各项估算之间的一致性较差时, 原因往往可以追溯到项目范围定义不充分或生产率数据不合适.

( ) 33-16. Empirical estimation models are typically based on 504

A. expert judgement based on past project experiences
B. refinement of expected value estimation
C. regression models derived from historical project data
D. trial and error determination of the parameters and coefficients

经验估算模型(empirical estimation models)通过分析历史项目数据, 采用统计方法(如回归分析), 建立项目特征与开发工作量之间的数学关系, 用于预测新项目的工作量、成本等. 其他选项涉及专家判断、估值方法细化或试错法, 不属于经验估算模型的核心.

(C) 16. 经验估算模型通常基于
A. 基于以往项目经验的专家判断
B. 对期望值估算的细化
C. **从历史项目数据中得出的回归模型**
D. 通过反复试验确定参数和系数

( ) 33-17. COCOMO II is an example of a suite of modern empirical estimation models that require sizing information expressed as: 505

A. function points
B. lines of code
C. object points
D. any of the above

COCOMO II 是一种现代软件成本估算模型, 支持多种软件规模度量方式, 包括功能点(function points)、代码行数(lines of code)、对象点(object points)等. 可以用这些不同方式表示项目规模, COCOMO II 都能进行估算.

(D) 17. COCOMO II 是一套现代经验估算模型的例子, 它要求以以下哪种方式表达规模信息:
A. 功能点
B. 代码行数
C. 对象点
D. **以上任意一种**

( ) 33-18. Putnam's software equation is a dynamic empirical model that has two independent parameters: a size estimate and an indication of project duration in calendar months or years. 506

Putnam's软件方程是一种用于软件项目估算的动态经验模型, 有两个独立参数: 软件规模估算(如代码行数或功能点)和项目工期(以日历月或年为单位). 通过这两个参数, 方程可以预测项目所需的工作量和进度.

(T) 18. 普特南(Putnam)软件方程是一个动态的经验模型, 它有两个独立的参数: 一个是规模估算, 另一个是项目日历月或年的持续时间指示.

( ) 33-19. Function points are of no user in developing estimates for object-oriented software. 507

功能点(Function points)是一种与开发方法无关的度量方式, 无论是传统开发方法还是面向对象方法, 都可以用功能点来估算软件规模和开发工作量, 因此功能点在面向对象软件开发中同样有用.

(F) 19. 功能点在为面向对象软件开发估算时没有用处.

( ) 33-20. In agile software development estimation techniques focus on the time required to complete each 508

A. increment
B. scenario
C. task
D. use-case

在敏捷软件开发中, 估算主要关注完成每个增量(increment)所需的时间. 增量是指可交付、可用的系统部分, 是逐步构建整个系统的基本单位. 敏捷方法(如Scrum)通常用迭代(sprint)来开发每个增量, 团队会评估完成当前增量所需的时间.

(A) 20. 在敏捷软件开发中, 估算技术侧重于完成每个所需的时间
A. **增量**
B. 场景
C. 任务
D. 用例

( ) 33-21. It is possible to use a modified function point technique to develop estimates for Web applications. 509

功能点法(Function Point)是一种常用的软件规模估算方法, 最初用于传统软件, 但可以针对Web应用的特点, 如页面数量、交互复杂度等, 进行调整. 经过修改后, 功能点法可用于Web应用的工作量和成本估算.

(T) 21. 可以使用改进的功能点技术对Web应用进行估算.

( ) 33-22. Using a statistical technique like decision tree analysis can provide some assistance in sorting out the true costs associated with the make-buy decision. 510

像决策树分析这样的统计技术, 可以系统列出各种可能的决策路径, 考虑每种选择下的成本、收益和风险, 帮助项目团队全面评估"自制"或"外购"的真实成本, 为决策提供量化依据.

(T) 22. 使用像决策树分析这样的统计技术可以在理清与自制-外购决策相关的真实成本方面提供一定的帮助.

( ) 33-23. Outsourcing always provides a simple means of acquiring software at lower cost than onsite development of the same product. 511

外包有时能降低成本，但并非总是比本地开发更便宜. 外包可能带来沟通障碍、质量控制难题、时差影响和隐藏费用等问题，也可能因需求变更或返工导致整体成本上升，因此外包并不总是比现场开发更简单、更便宜的获取软件方式.

(F) 23. 外包总是比在现场开发同样产品以更低的成本获得软件的简单方式.

( ) 34-1. Software projects are inevitably late and there is nothing that can explain why. 512

虽然软件项目经常延期，但并非不可避免或无法解释. 项目延期的原因包括需求变更、估算不准确、资源不足、沟通不畅等. 通过良好的项目管理和工程实践，可以分析和改进这些问题，因此不能说没有原因.

(F) 1. 软件项目不可避免地会延迟，并且没有任何可以解释原因的因素.

( ) 34-2. It is unethical to undertake a project that you know in advance cannot be completed by the customer's deadline, unless you inform the customer of the risk and establish a project plan that can deliver the needed system incrementally. 513

本题考查软件工程师的职业道德. 若明知无法按客户截止日期完成项目，却没有告知客户风险，也没有制定分阶段交付的计划，这属于不道德行为. 诚信和透明是软件工程师应遵守的职业规范，必须让客户了解真实情况，帮助其做出合理决策.

(T) 2. 如果你事先知道无法在客户的截止日期前完成项目，除非你告知客户相关风险并制定能够分阶段交付所需系统的项目计划，否则承接该项目是不道德的.

( ) 34-3. Which of the following is not one of the guiding principles of software project scheduling: 514

A. compartmentalization
B. market assessment
C. time allocation
D. effort validation

B选项"market assessment"(市场评估)不属于软件项目进度安排的指导原则. 进度安排的核心原则包括compartmentalization(分解任务)、time allocation(时间分配)和effort validation(工作量验证)，这些原则用于合理安排项目进度和资源. 市场评估是产品规划或需求分析阶段的活动，不属于进度安排的指导原则.

(B) 3. 以下哪一项不是软件项目进度安排的指导原则:
A. 模块化
B. **市场评估**
C. 时间分配
D. 工作量验证

( ) 34-4. Doubling the size of your software project team is guaranteed to cut project completion time in half. 515

在软件工程中，将团队人数加倍并不能保证项目完成时间减半. 团队规模扩大后，沟通、协调和管理的复杂度会增加，带来更多沟通成本和管理难题，可能影响效率. 布鲁克斯定律指出: "向进度落后的项目中增加人手，只会让项目更落后. "因此，团队人数和项目进度不是线性关系.

(F) 4. 将你的软件项目团队规模加倍，是否一定能将项目完成时间减半.

( ) 34-5. The software equation can be used to show that by extending the project deadline slightly 516

A. fewer people are required
B. you are guaranteed to meet the deadline
C. more lines of code can be produced
D. none of the above

软件方程(software equation)用于项目管理. 如果适当延长项目截止期限，完成同样工作所需人数会减少. 多给时间可以用更少的人完成项目. 延长时间不能保证一定按时完成(B错误)，也不一定能写出更多代码(C错误).

(A) 5. 软件方程可以用来说明，通过适当延长项目截止日期
A. **所需人数会减少**
B. 你可以保证按时完成截止日期
C. 可以产出更多的代码行数
D. 以上都不是

( ) 34-6. The 40-20-40 rule suggests that the least of amount of development effort be spent on 517

A. estimation and planning
B. analysis and design
C. coding
D. testing

40-20-40法则指软件开发项目的工作量分配大致为: 40%用于需求分析与设计，20%用于编码，40%用于测试和维护. 编码只占开发工作量的最少部分. 该法则强调，软件工程需要更多精力投入到前期分析设计和后期测试维护，以保证软件质量和项目成功.

(C) 6. 40-20-40法则表明，开发工作中最少的精力应花在以下哪一项上?
A. 估算与计划
B. 分析与设计
C. **编码**
D. 测试

( ) 34-7. A task set is a collection of 518

A. engineering work tasks, milestones, deliverables
B. task assignments, cost estimates, metrics
C. milestones, deliverables, metrics
D. responsibilities, milestones, documents

题目考查"任务集"(task set)的定义. 在软件工程中，task set是完成某项软件工程活动所需的一组具体工作内容，包含: 工程工作任务(engineering work tasks)，如需求分析、设计、编码、测试等; 里程碑(milestones)，即关键节点或重要阶段，用于跟踪进度; 可交付物(deliverables)，指完成任务后需要提交的成果，如文档、代码等. 选项A包含了这三项内容. 其他选项只包含部分相关内容，不够全面或准确.

(A) 7. 一个任务集是由以下哪些组成的
A. **工程工作任务、里程碑、交付物**
B. 任务分配、成本估算、度量指标
C. 里程碑、交付物、度量指标
D. 职责、里程碑、文档

( ) 34-8. The task (activity) network is a useful mechanism for 519

A. computing the overall effort estimate
B. detecting intertask dependencies
C. determining the critical path
D. specifying the task set to the customer

任务(活动)网络是图形化工具，用于表示项目各任务之间的关系. 它可以帮助发现任务之间的依赖关系(B)，也能用于分析和确定项目的关键路径(C)，即影响项目工期的最长路径. A选项(计算整体工作量)和D选项(向客户说明任务集)不是任务网络的主要用途.

(BC) 8. 任务(活动)网络是一个有用的机制，用于
A. 计算总体工作量估算
B. **检测任务间的依赖关系**
C. **确定关键路径**
D. 向客户说明任务集

( ) 34-9. Tasks that lie on the critical path in a task network may be completed in any order as long as the project is on schedule. 520

关键路径上的任务通常存在先后依赖关系，必须按照特定顺序完成，否则后续任务无法开始，项目整体进度会受到影响. 因此，关键路径上的任务不能随意调整顺序.

(F) 9. 任务网络中位于关键路径上的任务只要项目按计划进行，可以以任意顺序完成.

( ) 34-10. Two tools for computing critical path and project completion times from activity networks are 521

A. CPM
B. DRE
C. FP
D. PERT

CPM(Critical Path Method，关键路径法)和PERT(Program Evaluation and Review Technique，计划评审技术)是常用的网络分析方法，用于计算关键路径、关键活动、最早完成时间和项目总工期. DRE和FP与此无关.

(AD) 10. 用于从活动网络计算关键路径和项目完成时间的两种工具是
A. **关键路径法(CPM)**
B. 缺陷消除率(DRE)
C. 功能点(FP)
D. **计划评审技术(PERT)**

( ) 34-11. Timeline charts assist project managers in determining what tasks will be conducted at a given point in time. 522

时间线图(Timeline charts)能够清晰展示项目任务的起止时间和持续时间，帮助项目经理直观了解在任意时间点上进行或即将开始的任务，便于计划和协调.

(T) 11. 时间线图帮助项目经理确定在特定时间点将要进行的任务.

( ) 34-12. The best indicator of progress on a software project is the completion 523

A. of a defined engineering activity task
B. of a successful budget review meeting on time
C. and successful review of a defined software work product
D. and successful acceptance of project prototype by the customer

衡量软件项目进展的最佳指标是完成并成功评审一个明确的软件工作产品(如需求文档、设计文档、代码模块等). 只有这些产品被评审通过，才能真实反映项目的实际进展. 相比之下，单纯完成任务、开会或原型被客户接受，都无法全面客观地反映项目整体进度.

(C) 12. 软件项目进展的最佳指标是完成
A. 一个已定义的工程活动任务
B. 按时成功召开预算评审会议
C. **并成功评审一个已定义的软件工作产品**
D. 并被客户成功验收项目原型

( ) 34-13. Since iterative process model work best for object-oriented projects it is impossible to determine whether an increment will be completed on time or not. 524

迭代过程模型适用于面向对象项目，但这并不表示无法判断每个增量是否能按时完成. 通过合理计划、进度管理和持续跟踪，团队可以预测和控制每个增量的完成时间，因此"无法判断"这种说法是错误的.

(F) 13. 由于迭代过程模型最适用于面向对象的项目，因此无法确定某个增量是否能按时完成.

( ) 34-14. WebApp projects only require the creation of a macro schedule. 525

WebApp项目不仅需要宏观计划(macro schedule)，还需要微观计划(micro schedule). 宏观计划确定项目的重要里程碑和总体时间安排，微观计划则细化到每个具体任务、资源分配和进度跟踪，以确保项目顺利进行，因此只制定宏观计划是不够的.

(F) 14. WebApp项目只需要创建一个宏观进度表.

( ) 34-15. The purpose of earned value analysis is to 526

A. determine how to compensate developers based on their productivity
B. provide a quantitative means of assessing software project progress
C. provide a qualitative means of assessing software project progress
D. set the price point for a software product based on development effort

挣值分析(Earned Value Analysis, EVA)是一种项目管理技术, 用于定量衡量项目进度和绩效. 它通过比较已完成工作的预算成本、实际成本和计划成本, 帮助项目经理判断项目是否按计划推进, 是否存在超支或滞后. 其他选项与挣值分析的本质无关.

(B) 15. 成本效益分析的目的是
A. 根据开发人员的生产力决定如何给予报酬
B. **为软件项目进展提供定量评估的方法**
C. 为软件项目进展提供定性评估的方法
D. 根据开发工作量为软件产品设定价格

( ) 34-16. Earned value analysis is a technique that allows managers to take corrective action before a project crisis develops. 527

挣值分析(Earned Value Analysis, EVA)是一种项目管理技术, 通过将已完成工作的价值与计划和实际支出进行比较, 帮助项目经理监控项目进度和成本. 如果发现项目有偏差, 经理可以及时采取纠正措施, 避免问题发展成更严重的危机.

(T) 16. 挣值分析是一种使管理者能够在项目危机发生前采取纠正措施的技术.

( ) 35-1. Proactive risk management is sometimes described as fire fighting. 528

Proactive risk management(主动风险管理)强调在风险发生前进行预测和预防, 而fire fighting(救火式管理)是指风险发生后才被动应对. 两者属于相反的管理方式, 主动风险管理不能被称为fire fighting.

(F) 1. 主动风险管理有时被描述为救火.

( ) 35-2. Software risk always involves two characteristics 529

A. fire fighting and crisis management
B. known and unknown risks
C. uncertainty and loss
D. staffing and budget

软件风险包含不确定性(uncertainty)和损失(loss)两方面. 不确定性指事件是否发生无法确定; 损失指风险事件发生会造成的损害. 这两点是所有软件风险共有的特性. 其他选项描述的是风险管理中可能涉及的内容, 但不是风险本身的基本特征.

(C) 2. 软件风险总是涉及两个特征
A. 救火和危机管理
B. 已知和未知的风险
C. **不确定性和损失**
D. 人员配备和预算

( ) 35-3. Three categories of risks are 530

A. business risks, personnel risks, budget risks
B. project risks, technical risks, business risks
C. planning risks, technical risks, personnel risks
D. management risks, technical risks, design risks

软件项目的风险一般分为三类:项目风险(project risks)、技术风险(technical risks)和业务风险(business risks). 项目风险指影响项目进度、成本、资源等的风险; 技术风险涉及技术实现的可行性、技术难题等; 业务风险与软件是否满足业务需求、市场变化等相关.

(B) 3. 风险分为哪三类?
A. 业务风险、人员风险、预算风险
B. **项目风险、技术风险、业务风险**
C. 计划风险、技术风险、人员风险
D. 管理风险、技术风险、设计风险

( ) 35-4. Generic risks require far more attention than product-specific risks. 531

一般性风险(generic risks)通常已经在项目管理流程中有标准的应对措施, 而产品特定风险(product-specific risks)对当前项目影响更大、更直接, 需要更多关注和管理. 所以说一般性风险比产品特定风险需要更多关注是不正确的.

(F) 4. 通用风险比产品特定风险需要更多关注.

( ) 35-5. A risk item checklist would contain known and predictable risks from which of these categories? 532

A. product size
B. development environment
C. staff size
D. process definition

风险项清单包括已知和可预见的多类风险来源: A. 产品规模, 产品越大风险越高; B. 开发环境, 工具、硬件、软件环境变化可能带来风险; C. 人员规模, 团队人数和经验影响项目进展; D. 过程定义, 过程不清晰或不完善容易导致项目失控. 这四类都是风险清单常见的风险来源.

(ABCD) 5. 风险项清单会包含哪些类别中已知和可预测的风险?
A. **产品规模**
B. **开发环境**
C. **人员规模**
D. **过程定义**

( ) 35-6. Questions that should be asked to assess the overall project risk include: 533

A. Have top managers formally committed to support the project?
B. Are end-users committed to the project and proposed system being built?
C. Are requirement fully understood by development team and customers?
D. Does the proposed budget have time allocated for marketing?

选项A、B、C与项目顺利推进密切相关: A关注高层管理者是否正式承诺支持项目, 没有管理层支持, 项目资源和决策容易受阻, 风险高. B关注终端用户是否愿意参与并接受新系统, 用户不支持, 即使系统开发完成也难以应用. C关注开发团队和客户是否充分理解需求, 需求不明确或理解不一致, 项目容易返工或失败. D涉及预算中是否包含市场营销时间, 营销主要是产品推广环节, 与项目开发本身的风险关系不大.

(ABC) 6. 评估整体项目风险时应提出的问题包括:
A. **高层管理人员是否正式承诺支持该项目?**
B. **最终用户是否承诺参与该项目及所构建的系统?**
C. **开发团队和客户是否完全理解需求?**
D. 拟定的预算中是否为市场营销分配了时间?

( ) 35-7. Software risk impact assessment should focus on consequences affecting 534

A. planning, resources, cost, schedule
B. marketability, cost, personnel
C. business, technology, process
D. performance, support, cost, schedule

软件风险影响评估关注风险发生后对项目带来的后果. 选项D的performance(性能)、support(支持)、cost(成本)、schedule(进度)涵盖了软件项目成功的关键方面, 这些因素直接影响产品能否按时、高质量、预算内完成, 以及后续的维护与支持, 风险评估时应重点考虑. 其他选项虽涉及部分重要内容, 但不如D全面.

(D) 7. 软件风险影响评估应侧重于影响哪些后果
A. 计划、资源、成本、进度
B. 市场性、成本、人员
C. 业务、技术、过程
D. **性能、支持、成本、进度**

( ) 35-8. Risk projection attempts to rate each risk in two ways 535

A. likelihood and cost
B. likelihood and impact
C. likelihood and consequences
D. likelihood and exposure

风险预测(risk projection)主要评估每个风险的发生概率(likelihood)和发生后的后果(consequences). 后果指风险实际发生时对项目造成的具体影响和损害. "consequences"是风险管理中的标准术语, 和likelihood一起作为核心评价要素. 其他选项虽然相关, 但不如"consequences"准确.

(C) 8. 风险预测尝试以两种方式对每个风险进行评估
A. 可能性和成本
B. 可能性和影响
C. **可能性和后果**
D. 可能性和暴露

( ) 35-9. Risk tables are sorted by 536

A. probability and cost
B. probability and impact
C. probability and consequences
D. probability and exposure

风险管理中的风险表通常按照风险发生的概率(probability)和影响程度(impact)排序, 这样可以优先处理概率高且影响大的风险, 便于制定应对措施. impact指风险发生后对项目的负面影响.

(B) 9. 风险表按照以下方式排序
A. 概率和成本
B. **概率和影响**
C. 概率和后果
D. 概率和暴露

( ) 35-10. Individual team members can make their own estimate for a risk probability and then develop a consensus value. 537

团队成员可以分别对风险发生概率进行估算, 然后通过讨论达成一致的估值. 这种方法在软件工程风险管理中常用, 有助于减少个人偏见, 提高概率估算的准确性.

(T) 10. 各个团队成员可以自行估算风险概率, 然后达成一个共识值.

( ) 35-11. Which factors affect the probable consequences likely if a risk does occur? 538

A. risk cost
B. risk timing
C. risk scope
D. risk resources

B项"risk timing"指风险在项目进程中发生的时间, 不同阶段发生风险会导致不同后果; C项"risk scope"指风险影响的范围, 范围越大, 后果越严重, 所以B和C会直接影响风险发生后的可能后果. A项"risk cost"是对风险后果的量化, 不是影响后果的因素; D项"risk resources"是用于应对风险的资源, 影响的是应对措施, 不是风险后果本身.

(BC) 11. 哪些因素会影响风险发生时可能产生的后果?
A. 风险成本
B. **风险时机**
C. **风险范围**
D. 风险资源

( ) 35-12. The reason for refining risks is to break them into smaller units having different consequences. 539

风险细化的目的是将复杂、笼统的风险分解为更具体、易于管理和分析的小风险，以便更好地理解风险来源和影响，便于采取针对性措施. 分解后的风险通常是原风险的具体表现或组成部分，其后果与原风险一致，而不是具有“不同的后果”. 题目中的说法不准确.

(F) 12. 细化风险的原因是将其分解为具有不同后果的更小单元.

( ) 35-13. Effective risk management plan needs to address which of these issues? 540

A. risk avoidance
B. risk monitoring
C. contingency planning
D. all of the above

有效的风险管理计划需要包括风险的规避(risk avoidance)、风险的持续监控(risk monitoring)，以及为潜在问题制定应急预案(contingency planning). 这三方面都是风险管理的重要内容，都需要被纳入风险管理计划.

(D) 13. 有效的风险管理计划需要解决以下哪些问题?
A. 风险规避
B. 风险监控
C. 应急计划
D. **以上全部**

( ) 35-14. Risk monitoring involves watching the risk indicators defined for the project and not determining the effectiveness of the risk mitigation steps themselves. 541

风险监控不仅包括观察项目中定义的风险指标，还要评估已采取的风险缓解措施(risk mitigation steps)是否有效. 只关注风险指标而忽略缓解措施效果，会导致风险管理不完整. 风险监控应包括监控风险指标和评估缓解措施实际效果两个方面.

(F) 14. 风险监控包括关注为项目定义的风险指标，而不是确定风险缓解措施本身的有效性.

( ) 35-15. Hazard analysis focuses on the identification and assessment of potential hazards that can cause 542

A. project termination
B. schedule slippage
C. cost overruns
D. an entire system to fail

Hazard analysis(危害分析)主要关注识别和评估可能导致整个系统失效的潜在危害，而不是项目终止、进度延误或成本超支等管理问题. 它关注系统层面，由危险事件(如硬件故障、软件缺陷等)引发系统整体功能丧失或不可用.

(D) 15. 危险分析侧重于识别和评估可能导致什么的潜在危险
A. 项目终止
B. 进度延误
C. 成本超支
D. **整个系统失败**

( ) 35-16. Risk information sheets (RIS) are never an acceptable substitute for a full risk mitigation, monitoring, and management (RMMM) plan. 543

题目说“风险信息表(RIS)永远不能替代完整的风险缓解、监控和管理计划(RMMM)”，这是错误的. 在项目规模较小或风险较少时，RIS可以作为RMMM计划的简化版本使用，所以RIS并不总是不能替代RMMM计划.

(F) 16. 风险信息表(RIS)绝不能作为完整的风险缓解、监控和管理(RMMM)计划的替代品.

( ) 36-1. How much effort is typically expended by a software organization on software maintenance? 544

A. 20 percent
B. 40 percent
C. 60 percent
D. 80 percent

软件维护在整个生命周期中占大量工作量，包括修复缺陷、适应环境变化和改进性能等. 通常，维护工作约占整个软件生命周期工作量的60%左右，这说明除了开发阶段，还要重视后期维护.

(C) 1. 软件组织通常在软件维护上投入多少工作量?
A. 20%
B. 40%
C. **60%**
D. 80%

( ) 36-2. Software supportability is not concerned with either the provision of hardware or infrastructure. 545

软件的可支持性(supportability)不仅关注软件本身，还包括它能否在特定硬件和基础设施上正常安装、运行和维护. 如果没有合适的硬件或基础设施，软件的支持和维护会变得困难，所以硬件和基础设施的提供与软件可支持性密切相关.

(F) 2. 软件可支持性与硬件或基础设施的提供无关.

( ) 36-3. Business process reengineering is often accompanied by software reengineering. 546

业务流程重组(BPR)时，原有软件系统通常难以适应新流程，因此需要对现有软件进行再工程，如修改、优化或重构软件系统，以支持新的业务需求. 所以BPR和软件再工程经常同时发生.

(T) 3. 业务流程再造通常伴随着软件再造.

( ) 36-4. Which of the following is not an example of a business process? 547

A. designing a new product
B. hiring an employee
C. purchasing services
D. testing software

业务流程(business process)是企业为实现某一业务目标而进行的一系列相关活动，如设计新产品、招聘员工、采购服务等，这些是实际业务运作的一部分. 测试软件属于技术活动，是软件开发过程的环节，不直接属于企业常规的业务流程，因此不是业务流程的例子.

(D) 4. 下列哪一项不是业务流程的例子?
A. 设计新产品
B. 雇佣员工
C. 采购服务
D. **测试软件**

( ) 36-5. Business process reengineering does not have a start or end, it is an evolutionary process. 548

业务流程重组(Business Process Reengineering, BPR)是一种有明确开始和结束的活动. 企业通过BPR对现有业务流程进行根本性再思考和彻底再设计，以实现重大绩效提升. BPR通常是一次性的、革命性变革，而不是持续进化的过程，因此有明确的启动和结束时间点，不同于持续改进那样的演化过程.

(T) 5. 业务流程重组没有开始或结束，它是一个进化的过程.

( ) 36-6. Which of the following activities is not part of the software reengineering process model? 549

A. forward engineering
B. inventory analysis
C. prototyping
D. reverse engineering

软件再工程(software reengineering)包括对已有软件系统进行改进和重构的活动. 常见活动有: 逆向工程(reverse engineering)，分析现有系统并提取设计和需求信息; 库存分析(inventory analysis)，盘点和评估现有软件资产; 正向工程(forward engineering)，利用逆向工程得到的信息重新开发或改进系统.

原型化(prototyping)是快速构建系统原型用于需求获取和验证的方法，属于软件开发过程，不是软件再工程过程模型的组成部分.

(C) 6. 下列哪项活动不属于软件再工程过程模型的一部分?
A. 正向工程
B. 库存分析
C. **原型设计**
D. 逆向工程

( ) 36-7. Software reengineering process model includes restructuring activities for which of the following work items? 550

A. code
B. documentation
C. data
D. all of the above

软件再工程(Software Reengineering)是对已有软件系统进行改造和优化的过程，主要包括对代码、文档和数据的重组和改进. 代码重组是优化和重构源代码，提高可维护性和可读性. 文档重组是更新、完善或重写文档，使其与当前系统一致且易于理解. 数据重组是优化数据结构或数据库，提升数据管理和存取效率. 因此，重组活动不仅针对代码，也包括文档和数据.

(D) 7. 软件再工程过程模型包括对以下哪些工作项的重构活动?
A. 代码
B. 文档
C. 数据
D. **以上所有**

( ) 36-8. Which of the following is not an issue to consider when reverse engineering? 551

A. abstraction level
B. completeness
C. connectivity
D. directionality

A项“抽象级别”(abstraction level)是逆向工程要考虑的，因为需要从低层次代码提炼出高层次设计. B项“完整性”(completeness)也是关注重点，逆向工程要尽量还原完整的软件结构和功能. D项“方向性”(directionality)指信息提取的方向，从实现到规范，是逆向工程的核心特征. C项“连通性”(connectivity)一般不是逆向工程关注的问题，它多用于系统结构或网络设计中，因此不是逆向工程主要考虑的议题.

(C) 8. 以下哪一项不是在逆向工程时需要考虑的问题?
A. 抽象层次
B. 完整性
C. **连通性**
D. 方向性

( ) 36-9. Reverse engineering of data focuses on 552

A. database structures
B. internal data structures
C. both a and b
D. none of the above

反向工程中的数据反向工程主要关注数据库结构(如表、关系、约束等)和程序内部使用的数据结构(如数组、链表、对象等)，目的是理解和还原已有系统的数据组织方式. 因此，既包括A，也包括B.

(C) 9. 数据的逆向工程侧重于
A. 数据库结构
B. 内部数据结构
C. **a 和 b 都是**
D. 以上都不是

( ) 36-10. The first reverse engineering activity involves seeking to understand 553

A. data
B. processing

C. user interfaces
D. none of the above

逆向工程的第一个活动是理解系统的处理过程，即系统如何运作、数据如何被处理．只有先理解了处理逻辑，才能进一步分析数据结构和用户界面．所以processing(处理过程)是逆向工程首先需要理解的内容．
(B) 10. 逆向工程的第一步活动涉及理解什么?
A. 数据
B. **处理过程**
C. 用户界面
D. 以上都不是

(　) 36-11. Reverse engineering should proceed the reengineering of any user interface. 554

逆向工程的目的是理解和分析已有系统的结构和功能，提取系统的设计和需求信息．在对用户界面做再工程(如改造或优化)前，需先通过逆向工程了解原界面的实现方式和业务逻辑，这样再工程才能合理改进．因此，逆向工程是再工程的前提和基础．
(T) 11. 逆向工程应当在对任何用户界面进行再工程之前进行．

(　) 36-12. Which of these benefits can be achieved when software is restructured? 555

A. higher quality programs
B. reduced maintenance effort
C. software easier to test
D. all of the above

软件重构(restructuring)是在不改变软件外部行为的情况下，对内部结构进行调整和优化．重构后，软件结构更清晰、可读性更强，带来以下好处：A. 更高质量的程序：结构优化使代码更易理解和维护，减少出错机会．B. 降低维护工作量：结构良好的代码便于定位和修复问题，维护成本降低．C. 更容易测试：结构清晰、模块划分合理的软件更方便编写和执行测试用例，提高测试效率．因此，A、B、C三项好处都可以通过重构实现．
(D) 12. 当软件被重构时，可以实现以下哪些好处?
A. 更高质量的程序
B. 减少维护工作量
C. 软件更易于测试
D. **以上全部**

(　) 36-13. Code restructuring is a good example of software reengineering. 556

代码重构(code restructuring)主要是优化和改进代码结构，不改变其外部行为．软件再工程(software reengineering)涉及对整个软件系统的大范围改造，包括分析、设计、重构和重新实现．代码重构只是再工程的一个部分，不是完整的再工程实例．
(F) 13. 代码重构是软件再工程的一个很好的例子．

(　) 36-14. Which of these is not an example of data restructuring? 557

A. data analysis
B. data name rationalization
C. data record standardization
D. none of the above

数据重组(data restructuring)是指对数据的结构、命名、格式等进行规范和调整，便于管理和使用．选项B“数据命名规范化”和C“数据记录标准化”都是数据重组的操作．A“数据分析”是对数据内容进行理解和处理，不涉及数据结构的调整，因此不属于数据重组．
(A) 14. 以下哪一项不是数据重组的例子?
A. **数据分析**
B. 数据名称规范化
C. 数据记录标准化
D. 以上都不是

(　) 36-15. Forward engineering is not necessary if an existing software product is producing the correct output. 558

即使现有软件产品能产生正确输出，前向工程(Forward engineering)仍然有必要．前向工程不仅用于修复错误，还包括适应新需求、改进系统结构、迁移到新平台或扩展功能等．如果仅因软件当前正常工作而不进行前向工程，可能会错过系统优化和持续发展的机会，因此前向工程依然必要．
(F) 15. 如果现有的软件产品能够产生正确的输出，则无需进行正向工程．

(　) 36-16. Reengineering client/server systems begins with a thorough analysis of the business environment that encompasses the existing computing system. 559

对客户端/服务器系统进行再工程时，首先要对包含现有计算系统的业务环境进行全面分析．再工程的目标不仅是改进技术本身，还要确保新系统能够满足业务需求，适应业务流程．因此，必须先了解整个业务环境，才能有针对性地对系统进行改造和优化．
(T) 16. 对客户/服务器系统进行再工程的第一步是对包含现有计算系统的业务环境进行彻底分析．

(　) 36-17. The only time reengineering enters into work with a legacy system is when it components will be implemented as objects. 560

题目认为只有在把遗留系统的组件实现为对象时才会用到再工程，这是错误的．再工程(reengineering)不仅在实现为对象时适用，还包括对遗留系统进行改进、优化、重构等多种情况，比如提升系统的可维护性、性能和可扩展性，无论是否采用面向对象的方法．
(F) 17. 只有当遗留系统的组件将被实现为对象时，才会进行再工程．

(　) 36-18. The cost benefits derived from reengineering are realized largely due to decreased maintenance and support costs for the new software product. 561

重构(reengineering)让新软件产品的代码更清晰、结构更好、技术更先进，维护和扩展更容易，因此维护和支持所需的人力、时间和资金减少，实现成本节约．这也是软件重构常见的主要经济动机．
(T) 18. 从软件再工程中获得的成本效益主要是由于新软件产品的维护和支持成本降低．

本题目集由 @memset0 制作，如有问题可扫描右侧二维码在 CC98 留言反馈．

- **题面爬取自课程组官网，感谢 @小角龙 学长．**
- **使用 GPT 4.1 生成题目的中文翻译与解析．**
- **因原网站只支持单选，多选题通过增设 E 选项给出，现将其拆分为期末考形式的多选题．**