体系结构2019预测题目

必考

1. What distinguishes an architecture for a software product line from an architecture for a simple product?
   - 产品线的目的：实现高重用性，高可修改性
   - 产品线之所以如此有效，是因为可以通过重用，充分利用产品的共性，从而产生生产的经济性
   - 可重用的范围广
   - 在产品线架构中有**一组明确允许发生的变化**，然而对于常规架构来说，只要满足了单个系统的行为和质量目标，几乎任何实例都是可以的。因此，识别允许的变化是架构责任的一部分，同时还需要提供内建的机制来实现它们。
2. How to model quality attribute scenarios? Graphically model two quality attributes in "stimulus-response" format
   - availability
   - modifiability
   - testability
   - reusability
   - performance
   - usability
   - security
   - interoperability
3. Describe outputs generated from each phase of ATAM process.
   1. partnership & preparation 寻找合作伙伴、准备（实际从0开始）

      主要内容：分析架构文档，给出评估计划

      输入：架构文档

      输出：评估计划，包含：利益相关者的初步列表、评估的时间 地点 方式、评估报告该啥时候给出 给谁、评估报告中应该包含哪些内容
   2. evaluation 1

      过程
      - 评估组展示他们对项目的理解，以及一个预先评估的结果展示
      - 展示商业目标：负责人展示项目的商业相关内容：功能性需求 技术 管理 财务 政策限制 商业目标 主要利益相关者 主要质量属性
      - 架构组展示他们详细的架构设计：技术限制 系统交互 架构方法
      - 评估组识别架构方法(style pattern tactic)，可以总结生成目录
      - 评估组生成效用树
      - 根据效用树分析架构方法——到这边应该知道最重要的架构设计和……之间的关系

      输出

- 架构的详细展示
- 商业目标的关键点
- 以场景的形式描述的按优先级需求列表排序的质量需求
- 效用树
- 有风险决策和无风险决策
- 敏感点和权衡点

3. evaluation 2

   过程

   - 对利益相关者展示前一阶段的成果 1~6 不需要效用树了
   - 利益相关者根据自身角色进行头脑风暴，想出一些和个人相关的场景，并对场景进行优先级排序
   - 分析架构方法
   - 展示结果：架构文档 含有优先级的场景 效用树 敏感点和权衡点 有风险决策和无风险决策 风险主题及其一一对应的驱动因素

   输出

   - 从利益相关者那边拿到一份带有优先级的场景列表
   - 风险主题及其一一对应的驱动因素

4. follow-up 跟进

   - 让关键利益相关者评审，产出最终评估文档

   输出

   - 最终评估文档

【最终产出】：架构的详细展示 关键商业目标 通过质量属性场景描述的按优先级需求列表排序的质量需求 效用树 有风险决策和无风险决策 架构决策和质量需求的映射关系 敏感点和权衡点 评估报告

4. Map, and list four views of each category of style.
   - Allocation Style——系统与环境中的非软件结构的关系
     - 物理视图 部署视图 安装视图 开发视图 map-reduce work assignment
   - Component and connector Style——系统中元素的运行时行为和交互
     - Broker p2p SOA CS Pipe-and-filter publish-and-subscribe multi-tier
   - Module Style——系统是如何构建为一组执行单元（implementation unit）的
     - 逻辑视图Logical decomposition uses generalization layered aspect Data-Model 泛化视图 切面视图 分解视图 分层视图
5. Briefly describe the general activities in a software architecture process, and the major inputs and outputs at each activity.

   版本一： Architecture Activities

   - 为系统创建商业用例(场景)
   - 理解需求
   - 设计、选择架构
   - 对架构进行交流(利益相关者包括开发者)

- 分析、评估架构
- 实现架构
- 保证一致性

版本二： Architecture Process

- 识别 ASR
- 架构设计
- 架构文档化
- 架构评估

6. Explain the context, benefits and limitations of …… Pattern.
    -
7. architecture documentation
    1. What should be included in a typical software architecture documentation package? Briefly describe each component and its purpose.
        - 基本展示：展示组件和视图之间的关系，通常图形化展示
        - 组件目录：组件的详细信息、组件及其属性、关系及其属性、组件间的接口和行为（组件之间的相互关系 视图如何被归档）
        - 上下文图、上下文信息：展示整个系统和组件之间的关系
        - 可变性指导：如何从视图角度提高架构对可变性需求的满足
        - 合理性：有力的设计原因 设计理由
        - 接口和依赖
        - 限制
        - 测试用例和场景
    2. Why should a software architecture be documented using different views? Give the name and purposes of 4 example views.

        原因：

        - 不同的视图支持不同的目标和用户，突出不同的系统元素和关系
        - 不同的视图将不同的质量属性暴露出不同的程度

        视图：可能考1

可能考

1. Describe 4+1 view
    - 逻辑视图：描述架构级别的重要元素和他们在设计开发时的逻辑关系。

      对象或对象类 模块视图 概念类图
    - 进程视图：描述架构的并发和交流的元素，体现运行后的动态关系，比如数据流和控制流

      组件-连接器视图 顺序图
    - 开发视图：描述了开发环境中软件的静态组织结构，管理软件组件（一个管理工具）

      分配视图 构件图
    - 物理视图：描述主要的进程和组件是如何与应用硬件相映射的，反应了分布式特性

分配视图 部署图

- 架构用例：架构的描述可以围绕这四个视图来组织，再用一些用例或场景进行说明，形成了第五个视图

2. What are ASR? List four sources and methods for extracting and identifying ASRs.
   - ASR：Architecture Significant Requirements 系统中重要并且实现难度相对较大的质量需求
   - 来源：需求文档 与利益相关者的交流 对商业目标的了解 效用树
3. What are generic design strategies applied in designing software? Give a concise working example with software architecture for each strategy.
   - 分解：将复杂事务分解，实现每一部分
     - 把质量属性分解到单个组件上，满足约束要求，实现质量和商业上的需求
   - 抽象：突出显现问题
   - 逐步渐进、分而治之：逐步解决问题
   - 生成测试：生成测试，看是否符合需求
     - 利用已有的系统 初步的设计设计测试用例 不断迭代生成测试
   - 迭代：ADD
   - 重用element
4. Where do software architecture come from? List five possible sources of software architecture.
   - NFRs 非功能性需求
   - ASRs 系统中重要并且实现难度相对较大的质量需求
   - Quality Requirements 质量需求
   - Stakeholders 利益相关者
   - Organisations 组织
   - Technical Environments 技术环境
5. Describe relationships between architecture pattens and tactics. List four tactics names and describe their usage.

   模式 pattern vs 战术 tactic
   - 战术比模式简单
   - 模式把多个设计决策组合在一起
   - 都是架构师的主要工具
   - 战术是pattern设计的基石
   - 大多模式包含几个不同的战术。例如分层模式包含增加内聚，降低依赖的战术

   tactic name：
   - ping/echo
   - heartbeat
   - 软件升级
   - 增加内聚
   - 降低耦合

- 重写配置
- 自检（自我测试）
- 异常检测
- 延迟绑定
- 识别攻击者
- 控制输入
- 限制复杂度
- 减少单点失效

#学习/2018-2019第2学期