Background

实验要求

1. 由项目小组讨论，自行提出一个中等复杂规模的软件系统的需求，包括功能性需求和非功能性需求。
2. 对1中所给出的软件系统功能性需求进行需求用例分析，对非功能性需求运用质量属性场景技术进行非功能性需求分析。
3. 基于需求分析的结果，综合运用软件体系结构风格和模式、质量属性设计的课程知识，给出该软件系统的架构设计决策。
4. 运用ATAM体系结构评估方法，对设计决策进行评估，重点分析该设计决策中的敏感点、权衡点、有风险决策和无风险决策。
5. 按照最终的架构设计决策，实现该系统(编程语言可自由选择
6. 提交项目实验报告，内容包括项目组成员及分工、项目时间进度安排、软件需求分析、软件架构设计方案、项目组如何进行架构评估的过程描述和评估结果、软件实现及功能测试和非功能性测试。
7. 以项目小组为单位进行项目验收答辩，答辩环节包括
    1. 10~12分钟的项目实验汇报(汇报内容可参考项目实验报告
    2. 以及对实现的软件系统的现场演示讲解
    3. 授课教师和其他项目小组成员质询及回答

考核方式

教师评分占60%，其他项目小组评分平均分占40%

1. 提需求和所学知识的结合度(30%)
2. 软件规模大小，即提出软件系统需求本身的复杂程度(10%)
3. 软件需求、设计、评估、实现的过程完整度(30%)
4. 项目实验报告撰写、项目验收答辩、软件系统演示环节的文字、表达应变等综合素养(25%)
5. 质询及问题回答(5%)

Plan

初步计划找一个**已经做好的软件系统**进行实验要求的分析和评估

软件系统需要包含**功能性需求**和**非功能性需求**的解决方案

非功能性需求的介绍

- **性能（Performance）**：描述系统对资源的利用效率，包括响应时间、吞吐量、并发性等。例如，系统需要在高负载情况下保持稳定的性能。
- **可靠性（Reliability）**：描述系统在特定条件下的稳定性和可靠性，包括可用性、容错性、持久性等。例如，系统需要保证在长时间运行中不出现故障。
- **安全性（Security）**：描述系统对数据和资源的保护程度，包括数据加密、身份认证、访问控制等。例如，系统需要保护用户数据不被未授权的访问者获取。
- **可维护性（Maintainability）**：描述系统的代码结构、文档和测试等方面，以便于后续的修改和维护。例如，系统需要具有清晰的代码结构和详细的文档。
- **可扩展性（Scalability）**：描述系统在增加负载或规模时的性能表现，包括水平扩展和垂直扩展等。例如，系统需要能够容易地扩展以应对用户数量的增加。
- **可移植性（Portability）**：描述系统在不同平台或环境下的可移植性和兼容性。例如，系统需要能够在不同操作系统上运行，并且不受特定硬件或软件限制。
- **用户体验（Usability）**：描述系统的易用性和用户体验，包括界面设计、交互方式等。例如，系统需要具有直观的用户界面和简单易懂的操作流程。
- **可测试性（Testability）**：描述系统的测试性和可测度程度，包括测试用例的设计和执行等。例如，系统需要具有模块化的设计和良好的错误信息提示，以便于测试人员进行测试。

分析

需求用例分析

需求用例分析是一种将用户需求转化为系统功能性需求的方法。它主要通过描述用户与系统之间的交互来识别和定义功能需求。需求用例通常由以下几个要素组成：

1. **参与者（Actors）**：参与者是与系统交互的实体，可以是用户、外部系统或其他系统组件。
2. **用例（Use Cases）**：用例是描述系统对外部事件做出的反应，通常由参与者执行的动作和系统的响应组成。

3. **场景（Scenarios）**：场景是对用例的具体实例化描述，包括参与者的动作、系统的响应以及可能的交互流程。

用例分析通过分析各种可能的用户操作，将其转化为具体的用例和场景描述，从而帮助团队明确系统的功能需求。

质量属性场景技术分析

质量属性场景技术是一种用于分析和描述系统非功能性需求的方法。它主要通过场景描述系统在特定情境下的行为，以识别和定义系统的质量属性需求。质量属性场景通常由以下几个要素组成：

1. **场景（Scenarios）**：场景描述系统在特定情境下的行为，包括输入、操作、输出和可能的异常情况。
2. **质量属性（Quality Attributes）**：质量属性是系统在各种方面的表现和特性，包括性能、可靠性、安全性、可维护性等。
3. **情境（Context）**：情境描述场景发生的背景和环境，包括用户需求、系统配置、资源约束等。

质量属性场景技术通过分析系统在不同情境下的行为，识别和定义系统的质量属性需求，以帮助团队设计和实现满足用户期望的高质量系统。

ATAM体系结构评估方法

TAM（Architecture Tradeoff Analysis Method）是一种体系结构评估方法，旨在评估软件系统的架构设计，并识别其中的敏感点、权衡点以及潜在的风险和机会。以下是对ATAM体系结构评估的详细解释：

目的

ATAM的主要目的是评估系统架构设计的质量，并为软件项目提供决策支持。通过ATAM评估，团队可以识别架构设计中的优势和劣势，并提供改进的建议，以确保系统能够满足其预期的功能和质量需求。

过程

ATAM评估通常分为以下步骤：

1. **准备阶段**：定义评估的范围和目标，确定参与评估的利益相关者，并建立评估团队。
2. **评估阶段**：通过一系列的场景和质量属性来评估系统的架构设计。评估团队会根据这些场景和质量属性对系统进行分析，识别其优势和劣势。

3. **分析阶段**：评估团队对架构设计中的敏感点、权衡点、风险和机会进行深入分析。他们可能会使用工具和技术来帮助分析架构的各个方面，并识别其中的问题和潜在的解决方案。

4. **报告阶段**：评估团队撰写评估报告，总结评估结果并提出改进建议。这些建议可能包括对架构设计的调整、风险管理策略以及未来的改进方向。

关键概念

在ATAM评估中，有几个关键概念需要理解：

1. **敏感点（Sensitivity Points）**：指架构设计中可能影响系统性能和质量的关键决策点。

2. **权衡点（Tradeoff Points）**：指在不同质量属性之间存在的权衡和折衷情况。

3. **风险决策（Risk Decisions）**：指可能导致系统失败或性能下降的潜在风险。

4. **无风险决策（Non-Risk Decisions）**：指不会对系统性能和质量造成负面影响的决策。

价值

通过ATAM评估，团队可以全面了解系统架构设计的优势和劣势，及时发现和解决潜在的问题，从而提高系统的质量和可靠性。此外，ATAM评估还可以促进团队之间的沟通和协作，确保所有利益相关者对系统的架构设计都有清晰的理解和共识。

综上所述，ATAM是一种用于评估软件系统架构设计的方法，通过识别敏感点、权衡点和风险决策，帮助团队全面理解系统的架构设计，并提供改进建议，以确保系统能够满足其预期的功能和质量需求。