实验十一 软件体系结构设计（三）

实验目的：

1. 深入理解体系结构的设计和评估改进
2. 完成 SAD

实验内容：

1. 继续完成上周的实验任务 （有同学反映上周实验内容较多）

**KWIC 背景**

KWIC 索引系统是一种用于对文档中的关键字及其上下文进行索引和排序的系统，是软件工程中一个经典的案例。它包括几个主要功能模块，例如输入处理、旋转操作、排序操作、以及输出处理。KWIC 的体系结构风格可以影响系统的灵活性、可伸缩性和维护性。

**需求过程概述**

需求过程通常包括以下几个步骤：

（1） 需求获取

在需求获取阶段，软件工程师需要与用户、客户进行交流，以了解系统必须实现的功能以及各种约束条件。对于 KWIC 案例，以下是一些关键需求：

a) 能够对输入的文档进行关键字的识别和旋转。
b) 能够对旋转后的关键字进行排序。
c) 最终输出排序后的关键字及其上下文。

（2） 需求分析

需求分析需要对获取到的需求进行仔细分析和细化，确保每个需求都是明确、可行且可验证的。对于 KWIC 项目，需求分析可能涉及以下要点：

a) 输入和输出格式的详细说明。
b) 如何处理多文档输入。
c) 错误处理和边界条件。

（3） 需求定义

需求定义涉及将分析的需求记录在需求说明文档中，并且以一种用户和开发团队都能理解的形式表达。例如，KWIC 系统的需求定义可能包括以下几方面：

a) 功能需求：系统应该能够读取输入的文档，生成关键字的旋转组合，并对旋转组合进行排序，最后输出结果。
b) 性能需求：系统应该能够高效地处理大文档集，响应时间应在可接受范围内。
c) 用户界面需求：用户界面应该简洁易用，用户能够轻松输入文档并查看结果。

（4） 需求验证

需求验证阶段确保定义的需求正确且完整。可以通过评审、原型、或测试的方法进行验证。对于 KWIC 系统，可以通过以下方法验证需求：

a) 需求评审：与用户和利益相关者一起评审需求文档，确保所有需求准确反映用户的期望。
b) 构建原型：开发系统原型，供用户试用和反馈，测试关键功能是否满足需求。
c) 自动测试：编写测试用例，验证系统在各种输入条件下是否表现正确。

基于需求过程的分析，以及 KWIC 案例常见的体系结构风格，针对每种体系结构在下列属性上的表现进行打分，表格如下：

| 属性 | Priority | Pipe and Filter | Client-Server | Peer-to-Peer | Publish-Subscribe | Repository | Layered |
|---|---|---|---|---|---|---|---|
| 易于更改算法 | 1 | 5 | 3 | 2 | 4 | 1 | 3 |
| 易于更改数据表示 | 4 | 1 | 4 | 3 | 5 | 2 | 3 |
| 易于添加功能 | 3 | 5 | 3 | 4 | 4 | 2 | 3 |
| 性能良好 | 3 | 5 | 4 | 3 | 3 | 4 | 3 |
| 数据表示效率 | 3 | 1 | 4 | 3 | 4 | 5 | 3 |
| 易于复用 | 5 | 4 | 4 | 3 | 3 | 5 | 4 |
| 总分 | - | 62 | 72 | 57 | 79 | 67 | 72 |

通过对各个属性的分析和打分，发布-订阅（Publish-Subscribe）体系结构在 KWIC 案例中的表现最佳，其次是客户-服务器和分层的体系结构。

评分分析如下：

A. 管道和过滤器（Pipe and Filter）

管道和过滤器架构风格因为组件可替换性高，所以更改算法非常方便，但由于数据变更需要影响到每一个过滤器，因此易于更改数据表示的得分较低。插入新过滤器较为简单，所以添加功能这一点得分较高。由于多次 I/O 操作，导致性能受限，因此性能得分较高但不是满分。数据表示效率较低，因为数据需要在过滤器之间多次传输。过滤器组件可重用性高，因此易于复用得分较高。

B. 客户-服务器（Client-Server）

客户-服务器架构由于客户端或服务器端的算法修改非常方便，所以易于更改算法的得分较高。通过通信协议统一处理数据表示问题，因此易于更改数据表示得分高。尽管可以添加功能，但需要对客户端和服务器端进行一定的调整，所以该项得分中等。在通过负载均衡优化系统性能的情况下，性能得分较高。数据传输效率高使得数据表示效率得分高。由于模组化设计使组件的重用性较高，因此易于复用的得分也较高。

C. 对等网络（Peer-to-Peer）

对等网络架构中，各对等端独立存在，算法的修改涉及较多节点，使得易于更改算法得分中等。更改数据表示需要在多个对等节点中传播，导致效率降低，因此易于更改数据表示得分中等。导入新对等节点较为方便，所以易于添加功能得分较高。尽管网络负载较高，但性能依然在可接受的范围内，因此性能得分中等。数据处理效率一般，因此数据表示效率得分也是中等。由于实现复杂程度较高，导致易于复用得分中等。

D. 发布-订阅（Publish-Subscribe）

发布-订阅架构风格中，发布与订阅解耦，使得更改算法成本低，因此易于更改算法得分高。数据格式灵活使得易于更改数据表示得分高，直接添加新的订阅者也使得添加功能得分高。尽管广播机制性能有所限制，但总体表现不错，因此性能得分较高。数据处理高效使得数据表示效率得分高。而组件解耦保证了复用性，因此易于复用得分也较高。

E. 信息库（Repository）

信息库架构中更改算法较为困难，因为算法依赖于数据库结构所以得分较低。改变数据结构会影响广泛，因此易于更改数据表示得分中等。新功能的添加需同时调整数据库和客户端，使得易于添加功能的得分中等。数据库在处理大规模数

据查询时性能表现较好，因此性能得分较高。数据库结构设计良好，数据表示效率高，因此数据表示效率得分高。并且数据库结构通常具有高复用性，所以易于复用得分高。

F. 分层（Layered）

在分层架构中，修改算法可能会影响多个层次，因此易于更改算法得分中等。每层能够独立处理数据结构，使得易于更改数据表示得分高。通过添加新层能够方便引入新功能，得分较高。层与层之间通信开销较大，导致性能得分中等。多层的数据处理较为复杂使得数据表示效率得分中等。层与层之间解耦，复用性高，因此易于复用得分高。

2. 完成自己项目的 SAD

**下周五（含）前将体系结构设计文档 SAD 提交给相应的助教**

项目跟踪，建立能反映项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。