实验十　软件体系结构设计（二）

实验目的：
1. 体系结构风格和视图特点
2. 研究经典软件体系结构案例
3. 继续补充和修改自己项目的 SAD

实验内容：

1. 对比书上各种软件体系结构风格和视图特点，思考自己项目属于哪种设计风格?

网上搜索最新的软件体系结构资料，如 MVC、Kruchten 4+1 视图等。

常见风格分类

体系结构风格的形成是多年探索研究和工程实践的结果。一个良好和通用的体系结构风格往往是工程技术领域成熟的标志。经过多年的发展，已经总结出许多成熟的软件体系结构风格，例如：

- 数据流风格：批处理和管道/过滤器。
- 调用/返回风格：主程序/子程序、层次结构和 C/S。 面向对象风格。
- 独立部件风格：进程通信和事件驱动。
- 虚拟机风格：解释器和基于规则的系统。
- 数据共享风格：数据库系统和黑板系统。

体系风格详述

（1）管道/过滤器 体系结构风格

主要包括过滤器和管道两种元素。在这种结构中，构件被称为过滤器，负责对数据进行加工处理。每个过滤器都有一组输入端口和输出端口，从输入端口接收数据，经过内部加工处理之后，传送到输出端口上。数据通过相邻过滤器之间的连接件进行传输，连接件可以看作输入数据流和输出数据流之间的通路，这就是管道。

（2）面向对象 体系结构风格

在面向对象体系结构中，软件工程的模块化、信息隐藏、抽象和重用原则得到了充分的体现。在这种体系结构中，数据表示和相关原语操作都被封装在抽象数据类型中。在这种风格中，对象是构件，也成为抽象数据类型的实例。对象与对象之间，通过函数调用和过程调用来进行交互。

（3）事件驱动 体系结构风格

事件驱动就是在当前系统的基础之上，根据事件声明和发展状况来驱动整个应用程序运行。事件驱动体系结构的基本思想是：系统对外部的行为表现可以通过它对事件的处理来实现。在这种体系结构中，构件不再直接调用过程，而是声明事件。系统其他构件的过程可以在这些事件中进行注册。当触发一个事件的时候，系统会自动调用这个事件中注册的所有过程。因此，触发一个事件会引起其他构件的过程调用。

（4）数据共享 体系结构风格

数据共享风格也成为仓库风格。

在这种风格中，有两种不同类型的软件元素：一种是中央数据单元，也成为资源库，用于表示系统的当前状态；另一种是相互依赖的构件组，这些构件可以对中央数据单元实施操作。中央数据单元和构件之间可以进行信息交换，这是数据共享体系结构的技术实现基础。
根据所使用的控制策略不同，数据共享体系结构可以分为两种类型，一种是传统的数据库，另一种是黑板。
如果由输入流中的事件来驱动系统进行信息处理，把执行结构存储到中央数据单元，则这个系统就是数据库应用系统。
如果由中央数据单元的当前状态来驱动系统运行，则这个系统就是黑板应用系统。
黑板是数据共享体系结构的一个特例，用以解决状态冲突并处理可能存在的不确定性知识源。
黑板常用于信号处理，如语音和模式识别，同时在自然语言处理领域中也有广泛的应用，如机器翻译和句法分析。

（5）解释器 体系结构风格

解释器作为一种体系结构，主要用于构建虚拟机，用以弥合程序语义和计算机硬件之间的间隙。实际上，解释器是利用软件来创建的一种虚拟机，因此，解释器风格又被称为虚拟机风格。

最新的软件体系结构：

MVC:

MVC（Model-View-Controller）是一种常见的软件体系结构模式，用于组织和设计应用程序的代码结构。它将应用程序划分为三个核心组件：模型（Model）、视图（View）和控制器（Controller），每个组件负责不同的职责。
MVC 模式的优点包括：
分离关注点：将数据、展示和用户交互分离，使得各个组件可以独立地演化和测试。
提高可维护性：模块化的结构使得修改和扩展变得更加容易。
支持多种界面：通过控制器作为中间层，可以支持多种不同的视图实现。

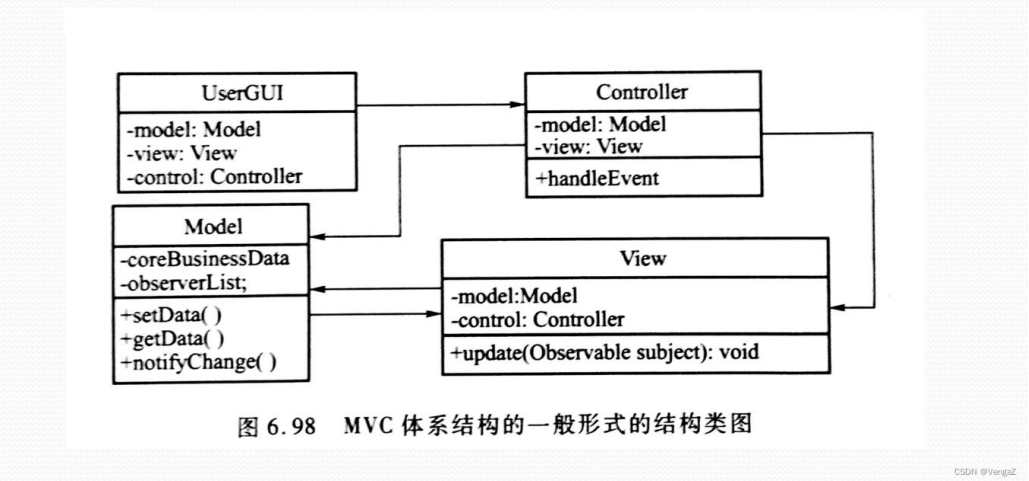


图 6.98　MVC 体系结构的一般形式的结构类图

Kruchten 4+1 视图

软件架构用来处理软件高层次结构的设计和实施。它以精心选择的形式将若干结构元素进行装配，从而满足系统主要功能和性能需求，并满足其他非功能性需求，如可靠性、可伸缩性、可移植性和可用性。Perry 和 Wolfe 使用一个精确的公式来表达，该公式由 Boehm 做了进一步修改：

软件架构 = {元素，形式，关系/约束}

软件架构涉及到抽象、分解和组合、风格和美学。我们用由多个视图或视角组成的模型来描述它。为了最终处理大型的、富有挑战性的架构，该模型包含五个主要的视图（请对照图 1）：

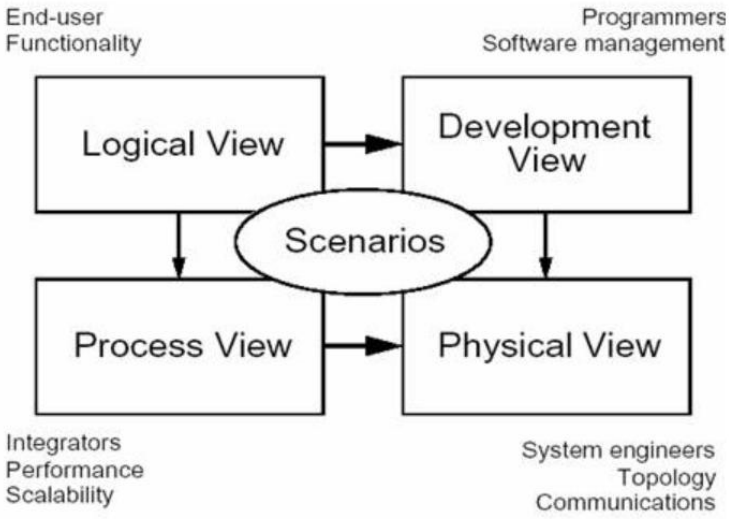
逻辑视图（Logical View），设计的对象模型（使用面向对象的设计方法时）。

过程视图（Process View），捕捉设计的并发和同步特征。

物理视图（Physical View），描述了软件到硬件的映射，反映了分布式特性。

开发视图（Development View），描述了在开发环境中软件的静态组织结构。

架构的描述，即所做的各种决定，可以围绕着这四个视图来组织，然后由一些用例 （use cases）或场景(scenarios)来说明，从而形成了第五个**+1**视图。



2. 参阅课本和网上资料，研究经典软件体系结构案例 KWIC。

An Introduction to Software Architecture，4.1 节

On-the-Criteria-To-Be-Used-in-Decomposing-Systems-into-Modules（Example System 1）

http://www.cs.cmu.edu/~ModProb/index.html

KWIC

KWIC（上下文中的关键词）索引系统接受一组有序的行；每一行都是一组有序的词，并且每一个词都是有序的字符集合。任何一行都可以通过反复地将第一个词移到行的末尾来进行"循环移位"。KWIC索引系统按照字母顺序输出所有行的循环移位列表。

共享数据解决方案

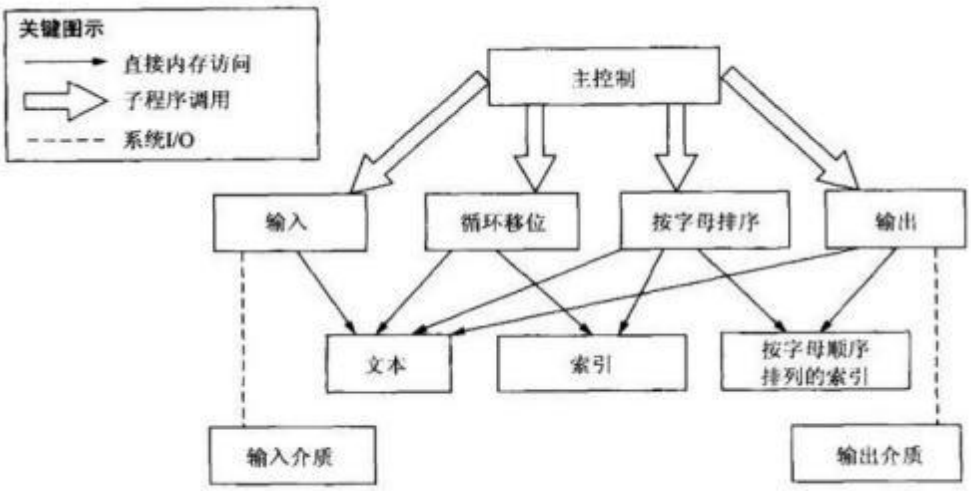


图5-13　KWIC的共享数据解决方案（Shaw and Garlan 1996）

数据模块解决方案

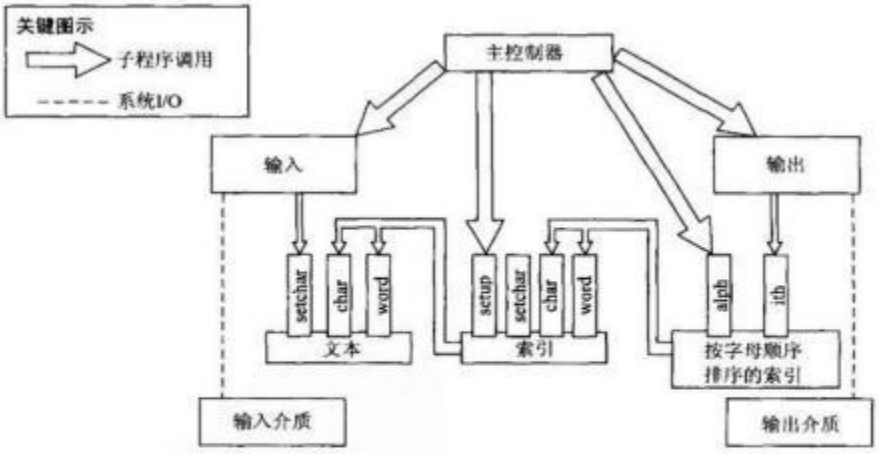


图5-14　KWIC的数据模块解决方案（Shaw and Garlan 1996）

另外，Garlan、Kaiser和Notkin给出了第三种方案（Garlan, Kaiser and Notkin 1992），如图5-15所

隐含调用解决方案

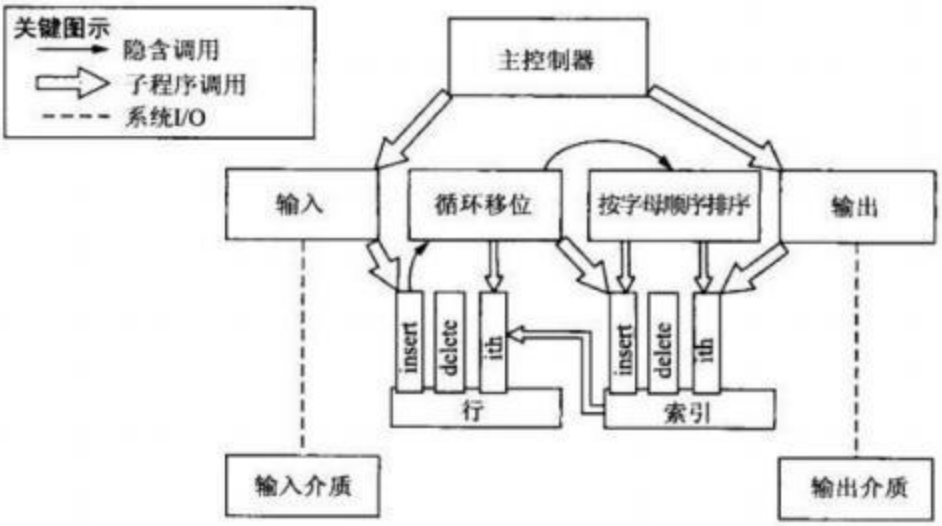


图5-15　KWIC的ADT解决方案（Shaw and Garlan 1996）

管道和过滤器解决方案

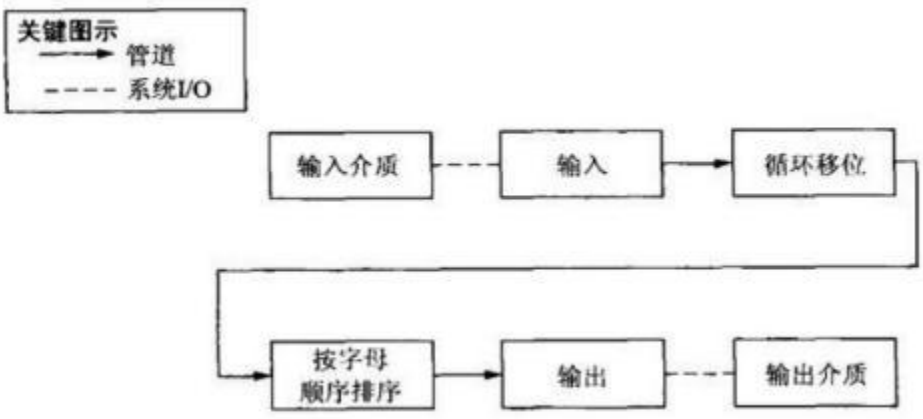


图5-16　KWIC的管道和过滤器解决方案（Shaw and Garlan 1996）

KWIC 方案带权重比较

属性	优先级	共享数据	数据抽象	隐含调用	管道和过滤器
易于改变算法	1	2	4	4	3
易于改变数据表示	4	2	3	3	2
易于改变功能	3	3	1	3	1
好的性能	3	2	2	3	2
有效的数据表示	3	1	4	3	2
易于复用	5	4	3	4	3
模块化	2	4	2	5	2
易测试性	3	3	2	1	1
安全性	1	1	2	3	4
易使用性	3	2	3	4	5
易理解性	3	1	3	1	3
易整合性	3	2	2	3	4

本项目方案带权重比较

属性	优先级	共享数据	数据抽象	隐含调用	管道和过滤器
易于改变算法	2	3	2	3	4
易于改变数据表示	4	3	3	2	3
易于改变功能	3	2	1	1	2
好的性能	5	3	2	3	2
有效的数据表示	3	4	3	4	3

易于复用	3	2	4	5	1
模块化	4	2	1	2	2
易测试性	4	2	4	4	3
安全性	5	3	2	2	2
易使用性	5	1	1	1	5
易理解性	5	3	3	1	4
易整合性	3	2	2	4	3

针对 KWIC 和自己项目，参考课本 ch5 表 5-3，小组成员每人给几种不同的体系结构风格设计打分，评最佳。

3. 补充和修改自己项目的 SAD

记录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。