Week9-4:

《On the Criteria To Be Used in Decomposing Systems into Modules》是由David L. Parnas于1972年发表的论文，被认为是软件工程领域的经典之作。本文探讨了将系统分解成模块的标准和准则，并提出了一种新的方法，以帮助软件开发者更好地组织和管理复杂系统。

该论文主要关注于系统分解成模块的标准和准则，以及模块化设计的原则，它提供了一些指导性的准则，帮助软件开发者更好地组织和管理复杂系统。它的主要贡献是引入了信息隐藏的概念，即将系统的内部细节隐藏在模块内部，仅通过公共接口与外部进行通信。这种方法可以实现模块之间的高内聚性和低耦合性，从而提高系统的可维护性、可重用性和可理解性。

论文提出了一些关于模块化的准则和原则，包括：

（1）抽象：将模块定义为一组相关功能的集合，并隐藏内部细节。通过定义清晰的接口，模块可以独立地开发和测试。

（2）信息隐藏：模块应该隐藏其内部实现细节，只通过公共接口与其他模块进行通信。这样可以减少模块间的依赖关系，提高系统的灵活性和可维护性。

（3）模块独立性：模块应该具有高内聚性和低耦合性。高内聚性意味着模块内的元素之间具有紧密的关联，低耦合性意味着模块之间的相互依赖性较低。

（4）接口设计：良好的接口设计是实现信息隐藏和模块独立性的关键。接口应该简单、清晰且易于理解，同时提供必要的功能。

（5）可重用性：通过将系统分解为模块，可以促进模块的重用，从而提高开发效率和系统的可维护性。

该论文的贡献对软件工程领域的发展产生了深远的影响。它强调了模块化设计的重要性，并提供了一些实用的原则和准则，帮助开发者更好地管理和组织复杂系统。这些原则和准则至今仍然被广泛应用于软件开发实践中，对提高软件质量和可维护性具有指导意义。

所以基于论文《On the Criteria To Be Used in Decomposing Systems into Modules》，我们可以分析一些经典的软件体系结构案例：

（1）Model-View-Controller (MVC) 架构模式

MVC 是一种常见的软件架构模式，它将一个应用程序分解成三个基本部分：模型（Model）、视图（View）和控制器（Controller）。模型是应用程序的数据和业务逻辑，视图是呈现数据的用户界面，控制器负责协调模型和视图之间的交互。MVC 架构模式具有高内聚性和低耦合性的特点，模块之间的依赖关系明确，易于维护和扩展。

（1）微服务架构模式

微服务架构模式是一种将应用程序分解成小型、自治的服务的方法。每个服务都是独立的、可替换的，并通过 API 与其他服务通信。微服务架构模式使得应用程序可以根据需要水平扩展，并提高了系统的可靠性和可维护性。每个服务都有明确的职责和接口，使得模块之间的依赖关系最小化，从而实现了高内聚性和低耦合性。

（2）分层架构模式

分层架构模式是一种将应用程序分解成层次结构的方法。每一层都具有明确的职责和接口，使得模块之间的依赖关系最小化。常见的分层架构模式包括三层架构模式和多层架构模式。三层架构模式将应用程序分解成表示层、业务逻辑层和数据访问层三个部分，多层架构模式将应用程序分解成多个层次，每个层次都具有明确的职责和接口。

这些经典的软件体系结构案例都采用了模块化设计的思想，具有高内聚性和低耦合性的特点，从而实现了良好的可维护性、可重用性和可扩展性。这些架构模式的应用在实际软件开发中已经得到了广泛的应用，并且对于实现高质量的软件系统非常有帮助。

这个网站提供了与软件架构中的模型有关的内容，包括模型的定义、分类、应用和评估等方面的内容。下面是一些有关网站中提供的模型的简要分析：

静态模型：静态模型主要关注软件系统的结构和组成部分之间的关系。网站提供了几种常见的静态模型，包括用例图、类图、对象图和包图等。

动态模型：动态模型关注软件系统的行为和交互方式。网站提供了几种常见的动态模型，包括顺序图、协作图、状态图和活动图等。

特定领域模型：特定领域模型是针对特定领域的软件系统而设计的模型。网站提供了几种常见的特定领域模型，包括数据流图、Petri 网络和有限状态机等。

架构模型：架构模型是描述软件系统整体结构和组成部分之间的关系的模型。网站提供了几种常见的架构模型，包括层次结构、客户端-服务器和分布式系统等。

质量模型：质量模型用于评估软件系统的质量，包括可维护性、可靠性、可用性和安全性等方面。网站提供了几种常见的质量模型，包括 ISO 9126 标准和 McCall 的质量模型等。

下面是该网站提供的一些具体项目的简要分析和学习：

关键字上下文（KWIC）

这是一个文本处理系统，通过给定一组行，创建一个按字母顺序排列的列表，这些行的旋转。KWIC 的核心目标是帮助用户快速定位和检索相关信息。这个系统的主要目标是对一组行进行处理，生成一个按字母顺序排列的列表，这些行是按照某种方式旋转的。这个系统的核心模型是一个流水线模型，它由三个主要组件组成：输入、旋转和输出。输入组件读取原始数据，旋转组件旋转每行，并将结果传递给输出组件。这个系统还可以视为一个基于过程的模型，其中每个过程都负责执行某个任务。例如，输入过程负责读取数据，旋转过程负责对行进行旋转，而输出过程负责将结果写入输出文件。

模型分析：

· 静态模型：类图用于表示系统中的实体，如行、旋转、排序等。

· 动态模型：顺序图可以描述处理行、旋转和排序的顺序。

· 架构模型：可以使用客户端-服务器模型，其中客户端向服务器请求操作，服务器执行操作并返回结果。

海浮标

海浮标系统自动在线收集和传输天气数据，满足紧急服务需求。这个系统的目标是自动和在线收集和传输天气数据，以满足抢占紧急服务的需求。这个系统的核心模型是一个分布式模型，其中每个海浮标都是一个节点，负责收集和传输数据。这个系统还包括一个中央节点，它负责收集和存储所有节点传输的数据，并将结果传递给用户。这个系统还可以视为一个事件驱动模型，其中海浮标节点产生数据事件，中央节点响应这些事件并执行必要的操作。

模型分析：

· 静态模型：类图表示浮标、传感器、数据收集器和传输器等组件。

· 动态模型：顺序图描述数据收集、处理和传输的过程。

· 特定领域模型：数据流图表示数据在系统中的流动。

· 架构模型：分布式系统模型，每个浮标都是一个独立的节点，汇报数据给中心节点。

巡航控制

巡航控制系统保持车辆速度在设定值。这个系统的目标是保持车辆的速度。这个系统的核心模型是一个反馈控制模型，其中车辆的速度被测量并与目标速度进行比较。如果实际速度低于目标速度，系统将增加油门，并监控速度的变化。如果实际速度过高，系统将减少油门，并再次监视速度的变化。这个系统还可以视为一个状态机模型，其中车辆的速度处于不同的状态，例如加速、匀速和减速状态。

模型分析：

· 静态模型：类图表示速度控制器、传感器、执行器等组件。

· 动态模型：状态图描述系统在不同速度情况下的状态转换。

· 特定领域模型：有限状态机显示系统的状态和转换。

· 架构模型：层次结构模型，将速度控制、传感器读取和执行器控制分为不同层。

会议征稿系统

用于会议征集、审稿和选择论文。这个系统的目标是为会议征集、审稿和选择论文。这个系统的核心模型是一个流程模型，其中每个阶段都由一个特定的角色执行。例如，提交论文的作者与审稿人和程序委员会成员交互，以确定哪些论文应该被接受或拒绝。这个系统还可以视为一个基于角色的模型，其中每个角色都有不同的权限和责任。

模型分析：

· 静态模型：类图表示论文、评审、主题等实体。

· 动态模型：顺序图描述提交、审稿和选择论文的过程。

· 架构模型：客户端-服务器模型，作者和评审作为客户端，服务器负责管理征稿过程。

邮件列表处理程序

合并多个来源的地址信息，消除重复并观察读者偏好。这个系统的目标是合并来自多个来源的地址信息，消除重复并观察读者偏好。这个系统的核心模型是一个数据流模型，其中地址信息流经多个处理器，每个处理器都执行特定的任务。例如，一个处理器可以将地址信息与已知的重复项进行比较，另一个处理器可以将地址信息与读者偏好进行比较。这个系统还可以视为一个基于事件的模型，其中每个事件都标识了新的地址信息，处理器响应这些事件并执行必要的操作。

模型分析：

· 静态模型：类图表示地址、来源、读者偏好等实体。

· 动态模型：活动图描述合并、消除重复和分析偏好的过程。

· 架构模型：层次结构模型，分为数据处理、分析和结果展示层。

打印机后台处理程序

管理打印机网络中的打印作业。这个系统的目标是管理打印机网络中的打印作业。这个系统的核心模型是一个队列模型，其中打印作业按照先进先出的顺序排队等待执行。这个系统还包括一个调度程序，它负责将打印作业分配给可用的打印机。这个系统还可以视为一个分布式模型，其中多个打印机可以同时执行打印作业。

模型分析：

· 静态模型：类图表示打印作业、打印机、队列等实体。

· 动态模型：协作图描述打印作业在打印机和队列之间的协作关系。

· 架构模型：客户端-服务器模型，客户端提交打印作业，服务器管理打印机网络。

图书馆系统

自动执行传统的图书馆任务，例如签入和签出书籍。这个系统的目标是自动执行传统的图书馆任务，例如签入和签出的书籍。这个系统的核心模型是一个基于过程的模型，其中每个过程都负责执行某个任务。例如，签入过程负责检查书籍是否已经签入，如果没有，则将其标记为签入，并将其放回书架上。这个系统还包括一个数据库，它负责存储图书馆的书籍和读者信息。

模型分析：

· 静态模型：类图表示书籍、读者、借阅记录等实体。

· 动态模型：活动图描述签入和签出书籍的过程。

· 架构模型：客户端-服务器模型，客户端为读者，服务器管理图书馆资源。

自动柜员机

通过远程定位提供通常的银行功能。这个系统的目标是通过远程定位提供通常的银行功能机器。这个系统的核心模型是一个交互模型，其中用户与机器交互，执行特定的任务。例如，用户可以使用机器提取现金、查询余额或转账。这个系统还包括一个网络，它负责将用户请求路由到正确的机器，并将结果返回给用户。

模型分析：

· 静态模型：类图表示账户、交易、银行等实体。

· 动态模型：顺序图描述ATM 与银行系统之间的交互过程。

· 特定领域模型：数据流图表示数据在 ATM 和银行系统之间的流动。

· 架构模型：客户端-服务器模型，ATM 作为客户端与银行后台服务器进行交互。

日历计划程序

组织会议日程。这个系统的目标是组织会议日程。这个系统的核心模型是一个日历模型，其中每个日历都表示某个用户的日程安排。这个系统还包括一个调度程序，它负责将多个用户的日程安排集成在一起，并计算可行的会议时间。这个系统还可以视为一个基于角色的模型，其中每个用户都有不同的日程安排和可用时间。

模型分析：

· 静态模型：类图表示事件、参与者、时间安排等实体。

· 动态模型：活动图描述创建、修改和删除事件的过程。

· 架构模型：客户端-服务器模型，客户端为用户，服务器负责管理事件和日程数据。

编译器

将编程语言的源代码转换为可执行文件。这个系统的目标是将编程语言的源代码转换为可执行文件形式。这个系统的核心模型是一个编译器模型，其中源代码被解析并转换为中间表示形式。然后，中间表示形式通过多个优化和转换阶段进行处理，直到最终生成可执行文件。这个系统还可以视为一个基于过程的模型，其中每个过程都负责执行特定的任务，例如词法分析、语法分析和代码生成。

模型分析：

· 静态模型：类图表示源代码、语法树、字节码等实体。

· 动态模型：活动图描述词法分析、语法分析、语义分析和代码生成的过程。

· 架构模型：层次结构模型，将编译过程分为不同阶段。

移动机器人

设计能够在监控的同时执行任务的移动机器人，例如避开障碍物。这个系统的目标是设计能够在监控的同时执行任务的移动机器人环境，例如避开障碍物。这个系统的核心模型是一个感知-决策-执行模型，其中机器人感知环境并根据其决策执行动作。例如，机器人可以通过传感器检测障碍物，并决定如何避开它们。这个系统还包括一个控制器，它负责将感知和决策组合起来，并将结果传递给执行器。

模型分析：

· 静态模型：类图表示机器人、传感器、执行器等组件。

· 动态模型：状态图描述机器人在不同状态下的行为。

· 特定领域模型：Petri 网络表示机器人的并行行为和资源分配。

· 架构模型：层次结构模型，将机器人的感知、决策和执行划分为不同层。