目录

[1.概述 1](#_Toc71744790)

[2.设计原则: 1](#_Toc71744791)

[2.1面向功能的分解 1](#_Toc71744792)

[2.2面向特征的分解 1](#_Toc71744793)

[2.3面向进程的分解 1](#_Toc71744794)

[2.4面向事件的分解 2](#_Toc71744795)

[2.5面向对象的设计 2](#_Toc71744796)

[3.常见体系结构视图 2](#_Toc71744797)

[3.1分解视图 2](#_Toc71744798)

[3.2依赖视图 2](#_Toc71744799)

[3.3泛化视图 2](#_Toc71744800)

[3.4执行视图 2](#_Toc71744801)

[3.5实现视图 2](#_Toc71744802)

[3.6部署视图 2](#_Toc71744803)

[3.7工作分配视图 3](#_Toc71744804)

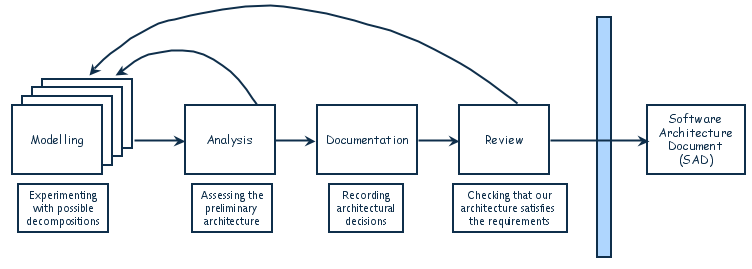
# 1.概述

设计是一种创造性的过程，它考虑如何实现所有客户的需求，设计所产生的计划也称为设计，早期的设计专注于系统的体系结构，后续的设计专注于如何实现单个的单元，设计软件是种具有智力挑战性的任务。通过学习优秀的设计例子可以提高设计技巧。

大部分设计工作是例程设计 (routine design), 通过对相似问题的解决方案进行复用和调整来解决当前问题.

# **2.设计原则:**

软件系统设计是一个迭代的过程,最终结果是软件体系结构文档 (SAD).



流行的设计方法:面向功能的分解、面向特征的分解、面向数据的分解、面向进程的分解、面向事件的分解、面向对象的设计。

## 2.1面向功能的分解

将功能或要求划分为模块

从要求规范中列出的功能开始

较低级别的设计将这些功能划分为子功能，然后分配给较小的模块

描述哪些模块（子功能）相互调用

## 2.2面向特征的分解

将功能分配给模块

高级设计从服务和功能集合的角度描述了该系统

低级别设计描述每个功能如何增强服务并识别功能之间的交互

## 2.3面向进程的分解

将系统划分为并发过程

高级设计：识别系统的主要任务、将任务分配给运行时间过程、解释任务如何相互协调

低级设计：更详细地描述了流程

## 2.4面向事件的分解

专注于系统必须处理的事件，并将事件责任分配给不同的模块

高级设计编目系统的预期输入事件

低级别设计将系统分解为状态，并描述事件如何触发状态转换

## 2.5面向对象的设计

将对象分配给模块

高级设计可识别系统的对象类型，并解释对象彼此的关系

低层设计详细说明了对象的属性和操作

# 3.常见体系结构视图

## 3.1分解视图

分解视图将系统描述为可编程单元

此视图可能是分层的

可能由多个模型表示

## 3.2依赖视图

依赖视图显示软件单元之间的依赖关系

此视图在项目规划中很有用

也可用于评估对某些软件单元进行设计更改的影响

## 3.3泛化视图

泛化视图显示的软件单元是相互概括或专业化的

此视图在设计抽象或可扩展的软件单元时很有用

## 3.4执行视图

执行视图是软件架构师绘制的传统箱箭图，从组件和连接器上显示系统的运行时间结构

每组件都是一个独特的执行实体，可能带有自己的程序堆栈

连接器是某些内部通信机制，例如通信通道、共享数据存储库或远程程序呼叫

## 3.5实现视图

实现查看地图代码单位到包含其实施的源文件

帮助程序员在源代码文件迷宫中找到软件单元的实现

## 3.6部署视图

部署将运行时间实体（如组件和连接器）映射到计算机资源（如处理器、数据存储和通信网络）上

它帮助架构师分析设计的质量属性，如性能、可靠性和安全性

## 3.7工作分配视图

工作分配视图将系统设计分解为可分配给项目团队的工作任务

帮助项目经理规划和分配项目资源，并跟踪每个团队的进度