第一章

软件危机是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致计算机软件的开发和维护过程中遇到的一系列严重问题。

软件体系结构层次：一般软件系统可分为系统、子系统、模块、类多个层次。其中，（系统——子系统——模块——类）层面一般称为体系结构层次。数据结构等称为编码层次。二进制代码称为可执行代码层次。

“4+1”视图模型从5个不同的视角包括：

逻辑视图、进程视图、物理视图、开发视图和场景视图来描述软件体系结构。

软件体系结构风格：管道过滤器风格、调用返回风格、正交与分层风格、共享数据风格。

第4章

质量属性包含：可用性、可靠性、性能、安全性、可修改性、易用性。

第6章

可用性战术根据其目的可分为：错误检测、错误恢复、错误预防

错误检测技术：ping 与 echo战术、heartbeat战术、Exception战术

错误恢复战术：Voting战术、主动冗余、被动冗余、备份、shadow操作、检查点/回滚

错误预防战术：从服务中删除战术、事务（一般概念）、事务（JTA）、事务（Spring）、进程监视器

第7章

可修改性战术可分为：局部化修改、防止连锁反映、推迟绑定时间

局部化修改是指在设计期间为模块分配任务，用以把预期的变更限制在一定的范围之内。一般具备以下几个子战术：维持语义的一致性、抽象通用的服务、预测期望的变更、泛化模块等。

推迟绑定时间战术：包括运行时注册、配置文件、多态、依赖反转（Ioc）、组件更换、遵守已定义的协议、面向切面编程（AOP）

-------------------------------------------------------------------------------

-----------------------------------------------------------------*下面是小道消息*

迪杰斯特拉认为软件应该注意（分解与结构化）

可修改性战术预测期望的变更，一个系统中可能进行修改的部分如下：

业务规则、对硬件、软件系统的依赖、非标准语言的特性、状态变量、模块间的交互、质量目标可能改变、运行环境。

软件架构要满足：修改性、易用性、性能等多种非功能属性。

经典的软件体系结构，包含了管道过滤器、调用返回、层次等体系结构形态，经典软件体系结构产生于软件规模较小的时代。诸如P2P、C/S基于SOA架构成为现代软件体系结构的代表。

软件体系结构的理想效果是可完整的、高效地重用整个软件体系结构，将现有体系结构应用于新的项目中。

对于程序员新手而言，在一个框架中进行开发，更像是在一个半成品的积木模型上进行再创造。

各个模块都为某个功能而串接，表现为模块间的依赖关系过于紧密，模块失去了相关的独立性。

仲裁者模块根据其所在位置以及所起的作用可以分为数据仲裁者和服务仲裁者。

典型的数据依赖性包括存储库、消息中间件等。

典型的服务仲裁者则是用于隔离服务之间由于接口等产生的依赖关系。

静态调度，是指在系统上线前实现离线分配相关资源的调度方法。

待续。。。。。。。。。。。。。。。。。。。。。