KnowleSys 软件设计指南

在 21 世纪，能获得成功的专业人士乃是理解商业的技术人员或者理解技术的商业人员。

0．架构设计

1．交互设计
2．组件设计
3．数据设计

4．代码设计
5．质量设计

架构设计

1. 目的：提供保证系统质量的构建蓝图
2. 原则：层次化，模块化，组件化
3. 指南：

系统需求=功能属性+质量属性

质量属性

客户：性能，安全，运行，使用，功能
开发：更改，移植，重用，集成，测试
商业：时间，成本，生命期

在大型系统中，质量属性更多地是由系统结构和功能划分来实现的，而不再主要依赖所选用的算法或数据结构

用途

架构是风险承担者进行交流的手段
架构是早期设计决策的体现
架构是可传递可重用的模型

交互设计

1. 目的：为用户提供舒适的软件使用体验
2. 原则：为用户着想
3. 指南：

概念思考	对于第一线的用户最有价值的东西是什么？
	例如：检查一个图表只是附带的任务，用户的真实目标是确定股价发展趋势，这意味着你完全不必去创建报表工具，而应当创建趋势监视工具。

行为设计	用户如何串连起各个软件元素来完成其目的？
	1. 控制权在用户 2. 直接性（形象，隐喻） 3. 一致性（动作，隐喻） 4. 宽容性 5. 反馈

界面设计	如何把提示信息和结果信息展示给用户？
	1. 可读性和流畅性 2. 结构和平衡 3. 形象显示元素间关系 4. 聚集和强调 5. 信息层次 6. 美学 7. 简洁（不要用词过多）

组件设计

1. 目的：提供用于架构系统的基本模块
2. 原则：对象化，接口化
3. 指南：

组件分类
技术组件，业务组件（工具与执行），界面组件

设计步骤
确定功能
确定输入要素
确定输出要素
封装处理逻辑
隐蔽内部信息
设计属性
设计方法
设计事件

数据设计

1．目的：提供完备而正确的数据模型

2．原则：实体化，关系化，完备化

3．指南：

步骤

设计概念数据模型 CDM

设计物理数据模型 PDM

设计存储过程 SP

设计性能调整方案 Index

实体分类

主题：业务，程序（界面，配置，统计）

规模：大、小

变化：递增、少变

建议

采用无意义主键

采用不活跃标志与删除标志

枚举码编辑标准化处理（利用对照表简化输入输出）

避免用触发器

用存储过程干重活

若要跨平台就避免用存储过程（封装在数据存取层 SQL 中）

代码设计

1．目的：提供软件的物理实现

2．原则：模块化，可读化

3．指南：

如何编写

化整为零

见名知义

先框架后代码

递增渐进

评审

标准风格

预防错误

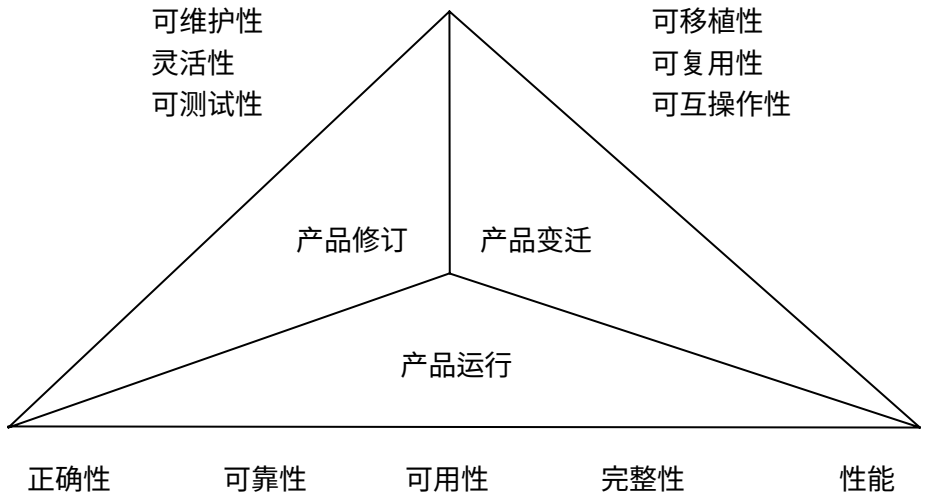
分离界面与功能

质量设计

1．目的：保证软件的运行质量和开发质量。

2．重点：可用性，可维护性，可复用性

3．指南：在结构设计时考虑质量属性



McCall 的软件质量因素