

# Review-软件工程核心思想

by:2022211928 gwc

## 软件工程的本质

(statement 1) 不同抽象层次之间的映射与转换

(statement 2) 用严格的规范和管理手段来缩小偏差, 通过牺牲“时间”来提高“质量”

任何软件系统开发的**共同本质**在于: 从**现实空间**的需求到**计算机(解)空间**的软件代码之间的**映射与转换**

**单步映射**与**多步映射**的**优缺点**分别都是什么?

## 软件工程的两个映射

- **概念映射**: 问题空间的**概念**与计算机解空间的模型化概念之间的映射
- **业务逻辑映射**: 问题空间的**处理逻辑**与计算机解空间处理逻辑之间的映射

例如: 学生成绩->概念 计算平均分->处理逻辑

**映射过程举例:**

### 问题空间

- (需求分析方法) --> 需求模型
- (系统设计方法) --> 设计模型
- (程序设计方法) --> 实现模型
- (部署与维护方法) --> 部署与运行模型

### --> 软件解空间

需求分析: 现实空间的需求-->需求规约

软件设计: 需求规约-->设计规约

实现: 设计规约-->代码

验证/确认

## 软件工程所关注的对象

软件工程具有“**产品与过程二相性**”的特点, 必须把二者结合起来去考虑, 而不能忽略其中任何一方

产品: 各个抽象层次的产出物

过程: 在各个抽象层次之间进行映射与转换

举例: **过程**-----**产品**

- |      |            |
|------|------------|
| 需求分析 | 系统用例图      |
| 系统分析 | BCE类图、分析类图 |
| 系统设计 | 设计类图       |
| 编码实现 | 源代码        |
| 软件测试 | 测试报告       |

## 软件工程所关注的目标

**1.功能性需求(Functional Requirements):** 软件所实现的功能达到它的设计规范和满足用户需求的程度

关注的目标:

- 完备性: 软件能够支持用户所需求的全部功能的能力
- 正确性: 软件按照需求正确执行任务的能力 (正确性描述软件在需求范围之内的行为)
- 健壮性: 在异常情况下, 软件能够正常运行的能力 (健壮性描述软件在需求范围之外的行为)
- 可靠性: 在给定的时间和条件下, 软件能够正常维持其工作而不发生故障的能力

**2.非功能性需求(Non-Functional Requirements):** 系统能够完成所期望的工作的性能与质量

关注的目标:

- 效率: 软件实现其功能所需要的计算机资源的大小, “时间-空间”
- 可用性: 用户使用软件的容易程度, 用户容易使用和学习 (当系统不再提供某种服务? --可用性下降)
- 可维护性: 软件适应“变化”的能力, 系统很容易被修改从而适应新的需求或采用新的算法、数据结构的能力 (维护什么? 何时维护? 谁来维护? 维护的效率? 代价?)
- 可移植性: 软件不经修改或稍加修改就可以运行于不同软硬件环境 (CPU、OS和编译器)的能力
- 清晰性: 易读、易理解, 可以提高团队开发效率, 降低维护代价
- 安全性: 在对合法用户提供服务的同时, 阻止未授权用户的使用 (抵抗/检测攻击、从攻击中恢复)
- 兼容性: 不同产品相互交换信息的能力
- 经济性: 开发成本、开发时间和对市场的适应能力
- 商业质量: 上市时间、成本/受益、目标市场、与老系统的集成、生命周期长短等

## 3.不同目标之间的关系--折中

- 不同类型的软件对质量目标的要求各有侧重:
  - 实时系统: 侧重于可靠性、效率
  - 生存周期较长的软件: 侧重于可移植性、可维护性
- 多个目标同时达到最优是不现实的:
  - 目标之间相互冲突

## 软件开发中的多角色

在软件开发过程中同样需要多种角色之间紧密协作, 才能高质量、高效率的完成任务

- 客户单位 (client, 甲方)
  - 决策者(CXO)
  - 终端用户(End User)
  - 系统管理员(Administrator)

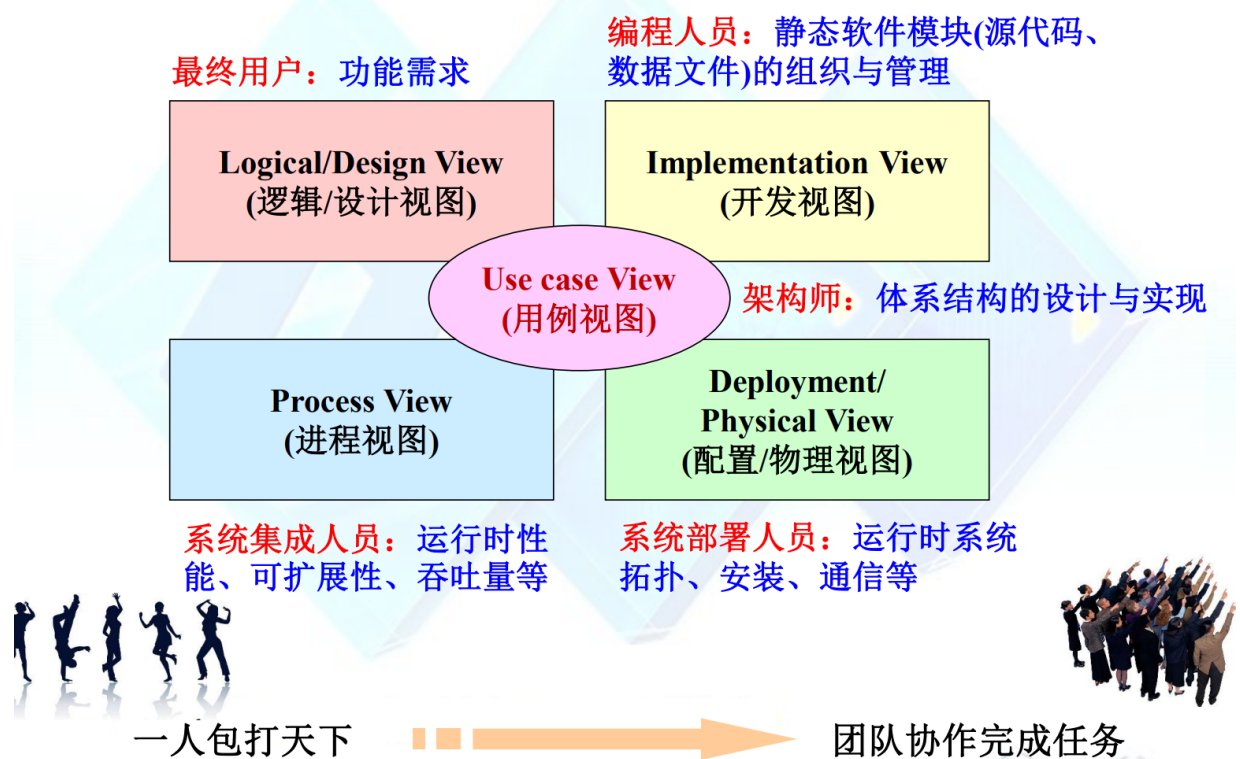
- 软件开发公司 (supplier, 乙方)
  - 决策者(CXO)
  - 软件销售与市场人员
  - 咨询师、需求分析师
  - 软件架构师、软件设计师
  - 开发人员：开发经理/项目经理、程序员
  - 维护人员

#### 不同的角色，视角不同，需求各有不同

不同的角色，他们所关心的非功能需求都有哪些？

不同角色的关注点之间，是否有重叠的情况？不同角色的关注点之间，是否有冲突的情况？

### 视角不同，需求各有不同



#### 软件工程=最佳实践

why?

软件系统的复杂性、易变性、不可见性使得高深的软件理论在软件开发中变得无用武之地

软件工程被看作一种**实践的艺术**

当你把所有的错误都犯过之后，你就是正确的了！

但在你自己展开实践之前，别人的任何经验对你来说都是概念——抽象、空洞、无物！

-FGX

最佳实践的例子：（省略）

## 软件工程四个核心理论概念

- **分而治之**（核心问题：如何的分解策略可以使得软件更容易理解、开发和维护？）
- **复用**（Don't reinvent the wheel ! Don't Repeat Yourself! ）
- **折中**（核心问题：如何调和矛盾（需求之间、人与人之间、供需双方之间、...））
- **演化**（核心问题：在设计软件的初期，就要充分考虑到未来可能的变化，并 采用恰当的设计决策，使软件具有适应变化的能力）