1、设计思维融合

把层、功能模块、细粒度模块三个概念分清楚。

- 功能模块是粗粒度的，一般对应一个功能组，最大的用途是基于功能模块进行开发小组分工。
- 层也是粗粒度的，UI交互层封装人机交互、系统交互层封装硬件访问和外部系统交互、数据管理层封装DB和File和Flsh存储。
- 模块划分设计需要进一步设计到细粒度模块一级。
    - 一个细粒度模块，必然位于架构的某一层中。
    - 一个细粒度模块，一般也都会属于某个更大粒度的功能模块内。
    - 如果架构设计“止于功能模块划分”或者“止于粗粒度分层”，则以为着设计不足、还要继续。

2、封装驱动设计方法-EDD

包含4个步骤：

- 研究需求。
    - 对模块划分促进最大的需求成果是上下文图和功能树。
    - 带着质疑的心态去研究、评价、确认，一旦发现问题，第一时间提出。
- 粗粒度分层
- 细粒度划分模块。
- 用例驱动的模块划分结构评审、优化。
    - 系统难点部分的设计，可多使用用例驱动方法。
    - 场景和用例，也是驱动设计评审的核心思维。

3、细粒度模块的划分技巧

3.1、分层细化

在粗粒度层内进一步“封装”职责形成更细致的层次结构，称之为sub-layer。

3.2、分区

将层内支持不同功能组的职责分别“封装”到“细粒度模块”中，从而使每个“粗粒度功能模块”由一组位于不同“层”的“细粒度模块”组成。

3.3、通用模块的分离

通用模块的分离，有利于提高重用性、减少重复代码。

- 高层模块应有较高的扇出，低层模块特别是底层模块应有较高的扇入。
- 扇入越大，表示该模块被更多的上级模块共享。
- 运用参数化、抽象等设计手段，提炼通用模块，供其他多个模块调用，这就避免了程序的重复。

3.4、通用机制的框架化

为了追求重用所带来的价值量最大化，将容易变化的部分封装成扩展点，并辅以回调机制将它们纳入框架的控制范围之内，从而在兼顾定制开销的同时使被重用的设计成果做多。

3.5、总结

- 经典的3层架构、4层架构，投标可能够用，但指导后续详细设计和并行开发绝对不够，还需要通过引入Sub-layer进行【分层的细化】。
- 根据功能树划分粗粒度的“功能模块”（纵切）和现有分层架构（横切）并不矛盾，为了引入这一维度的切分，还需要【分区】。
- 为了将软件系统的通用部分和专用部分相分离，便于重用或提炼程序库（Library）或框架（Framework），还需要【通用模块的分离】和【通用机制的框架化】。

4、优缺点

4.1、优点

- 更严谨、更专业，可操作性强。
- 综合了多种手段，促进了细粒度模块划分，利于关注点分离和模块复用。

4.2、缺点

- 因为是综合方法，所以掌握起来还是稍有门槛的。