道哥的第 027 篇原创

一、软件架构设计的生命周期

什么是架构？如果你问十个人，有可能得到十一种不同的答案；如果去翻一下相关的书籍，每一本都可能给出不同的定义。

因此，我们没必要纠结于那些概念，只要方法对、能完成项目任务就行，不管黑猫白猫，能抓到耗子的就是好猫！

1. 软件开发流程

一个软件项目，从立项开始到最终的交付，中间经历了诸多环节。具体到软件设计的生命周期来说，可包括这些阶段：需求调研-需求分析-概要设计-详细设计-架构验证-开发-单元测试-集成测试。

上面这几个步骤，仍然是非常粗略的概念。在实际操作中，其中的每一个步骤又可以细分为很多具体的执行环节。

例如：

> 1. 需求调研：应该怎么做？用什么方法？有什么指导思想？有什么好的工具？
> 2. 需求分析：应该怎么分析？所有的需求都应该被分析吗？如何找出关键需求？
> 3. 详细设计：在软件工程中有什么实践性比较好的方法？如何分层？如何分模块？

以上这些都是软件设计的过程中，需要涉及到的问题。那么软件设计的最终目标是什么呢？就是下面几个文档：

> 1. 逻辑架构；
> 2. 物理架构；
> 3. 运行架构；
> 4. 开发架构；

2. 关于套路

我认为，在这个世界上，一切皆有套路，包括任何事情、任何领域、任何行业。

当我们进入一个新的领域时，比如：让你设计一个车辆调度系统、机器人控制系统，或者设计一个对讲机、一个物联网网关，如果你是这个领域的新人，那么肯定是两眼一抹黑：我对这个领域完全不懂，怎么设计啊？

这让我想起一个小故事：

> 有一次我刚入职一家新公司，接手一位离职同事手里的工作。当时执行 KPI 考核，bug 直通率(就是一次性把 bug 解决掉的比例，QA 人员不会再把 bug 踢给你)，是一个重要的指标。面对系统里那么多的bug，领导问我：这些问题你大概需要多久能解决掉？我说：以前没接触过这方面的工作，没法给出准确的时间。领导说：没关系，你先给我一个具体的时间就行了。当时我就懵逼了。

在这个时候，最重要的事情就是，快速把这个领域里的基本的、重要的背景知识了解、掌握。那么应该如何做呢？找套路！

不要贪大求全，不要奢望把所有相关的内容都掌握，这是不可能的，尤其是在短时间内。我们的目标是把活做好，把项目完成。

这个时候，我一般的做法是：找套路！

这么说可能有点虚幻，那么就以软件开发中的架构设计来举例。在软件工程或者项目管理的书籍、资料中查找下面这些相关内容：

> 1. 别人是怎么来设计架构的？
> 2. 设计过程中需要哪些步骤？
> 3. 每一个步骤中，输入是什么？输出是什么？
> 4. 每一个步骤中，需要考虑的点是什么？
> 5. 有哪些好的软件工具？
> 6. 如何与项目的相关人进行沟通(项目经理、开发人员、测试人员、甲方客户)？

把以上的这些别人的经验进行梳理，总结出一套适合自己的“方法论”，然后在具体执行的时候按照这个套路一步一步的走，根据实际情况适时的动态调整，一般来说都能够顺利的推进一个项目。

3. 先僵化，后优化，再固化

这九个字是华为的掌舵人任正非在引进管理体系时提出的，这是一种非常实用的方法。

> 1. 僵化：站在巨人的肩膀上：处于学习初期阶段的“削足适履”；
> 2. 优化：掌握自我批判武器：在实践中不断吸收、改良、创新，优化自己；
> 3. 固化：创新是有阶段性的、受约束的，如果没有约束，创新就是杂乱无章、无序的创新，需要像夯土一样，一层层夯上去，一步步固化阶段性的优化成果；

对于软件架构设计，我们也可以按照这样的步骤走。

第一步就是僵化，也就是按照按照固定的套路走，虽然可能会“落入俗套”，但是至少能保证在正确的道路上，不会走歪，这是最重要事情！

如果被别人认为是“落入俗套”了，说明什么？说明别人已经认为你的做法符合这个领域中最“一般的”的做事流程，也就是相当于认可你已经真正进入这个领域了，这是好事情！

作为初学者，被别人这样评价，不应该觉得自豪吗？毕竟我们自己心里知道：我就是刚刚进入这个领域的小白而已。

4. 佛说："知我说法，如筏喻者"

我写这篇文章的目的，主要就是想聊一聊我是如何"僵化"的来进行软件架构设计的。

我们在刚进入软件开发行业的时候，每天的工作主要是撸代码，也许还不够资格来进行整个系统的架构设计。但是这并不妨碍我们自己来主动学习，机会都是留给有准备的人的，如果有一天，项目组里出现了架构设计的这个坑，那么领导会找哪个萝卜来填坑呢？

因此，这篇文章主要介绍的就是最初级、最基本的软件设计步骤，以及在每一个步骤中使用的指导思想、顺手的工具等等。

如果您已经是资深软件设计工程师，那么可以去喝咖啡、享受人生了，当然也可以分享您的方法论，我们相互学习！

这些内容也不是我自己摸索出来的，而是在项目开发过程中翻阅书籍、搜索资料、总结而来的，比较适合我所面对的项目，我也仅仅是知识的搬运工。

之前，我主要参考了几本关于软件架构设计的书籍，相互借鉴，总结出适合我的一套方法。上次搬家，好多书籍丢掉了，只剩下电脑里的零碎笔记了。这篇文章的素材来源，就是这些笔记，当然，大部分内容都是来源于书本，我只是进行了一些取舍、裁剪而已。

《金刚经》第六品，如来常说："汝等比丘，知我说法，如筏喻者；法尚应舍，何况 非法。"

佛法就像渡人过河的木筏，过了河上了岸，就应该把木筏丢掉，心中不要再想着木筏。木筏就像佛法，佛法尚且应该丢弃，何况不是佛法的东西，你还纠结他干嘛。

所以，我这里所讲的设计套路，也类似于木筏，用来帮助你在刚开始进入软件架构设计时的一个脚手架。当你进入到第二个段位"优化"时，就可以把这个脚手架扔掉了。

到那个时候，你就可以自信的说一句："道哥写的都是写什么东西啊，小儿科的东西，我需要向更高级的段位去探索"。到了这个时候，我就要衷心的恭喜你了！

二、需求调研和需求分析

大多数人都认同"需求决定架构"，但是需求究竟是"如何决定"架构的？这部分就来聊一下我的认识。

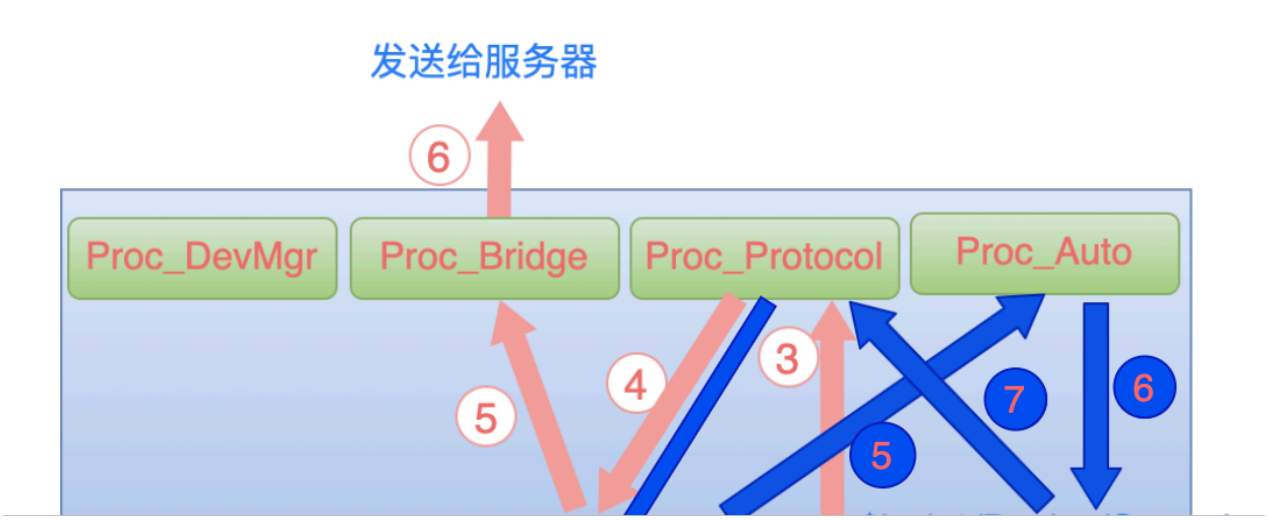
在立项阶段，如果运气比较好，可能会拿到一份文档《软件功能需求规格书》(话说一些日本公司的需求说明书，写的真的是变态的详细)。如果运气不好，什么文档都没有，所有的需求全部需要自己来收集、整理。

从狭义上来说，需求就是功能；从广义上来说，还包括质量属性、条件约束这些非功能需求。

1. 功能需求

功能需求部分是最直观的，就是我们设计的软件需要完成哪些事情。在一个系统中，各个功能之间不可能有清晰的边界的，每一个小模块之间通过一定的交互，形成一条一条的"协作链条"来完成指定的功能。

例如下面这张图，是我之前写的一篇文章：物联网网关开发：基于MQTT消息总线的设计过程(上) 中，不同模块之间的交互模型，红色和蓝色部分就是 2 条不同的协作链条。



当然了，这幅图是最终设计出来的系统架构(分层、分模块)，在得到这幅图之前，我们首先要把所有的功能需求进行收集、整理。

在这个阶段，最重要的事情是做什么，而不是怎么做。另外，作为设计人员，需要经常问自己一个问题：掌握的需求全不全？有没有遗漏？

2. 质量属性

我们可以把质量要求，按照不同的阶段来进行归类：

开发阶段

> 1. 可重用性，不要做重复的工作；
> 2. 灵活、容易扩展(想一想发生的需求变更时开发人员的心理活动)；

3. 容易理解(想想你接手别人的项目时);
4. 方便测试(单元测试、集成测试);
5. 可移植(尤其是嵌入式项目，需要运行在不同过的平台);

运行阶段

1. 系统必须可靠;
2. 性能必须达到一定要求(吞吐量、响应时间等等);
3. 没有漏洞，系统安全;
4. 伸缩性要好，方便扩展部署;

3. 条件约束

约束主要是指一些限制条件，比如:

1. 团队人员的技术栈情况(如果大家都用C，你就不要选择C++);
2. 领导给的资源有哪些;
3. 如果利用一些开源软件，是否存在bug？是否方便二次开发?
4. 项目开发周期是多久?
5. 软件的运行平台有哪些？有什么限制?

4. 画用例图

关于用例图的概念，我也总结不好，这里就直接引用百度知道里的定义了:

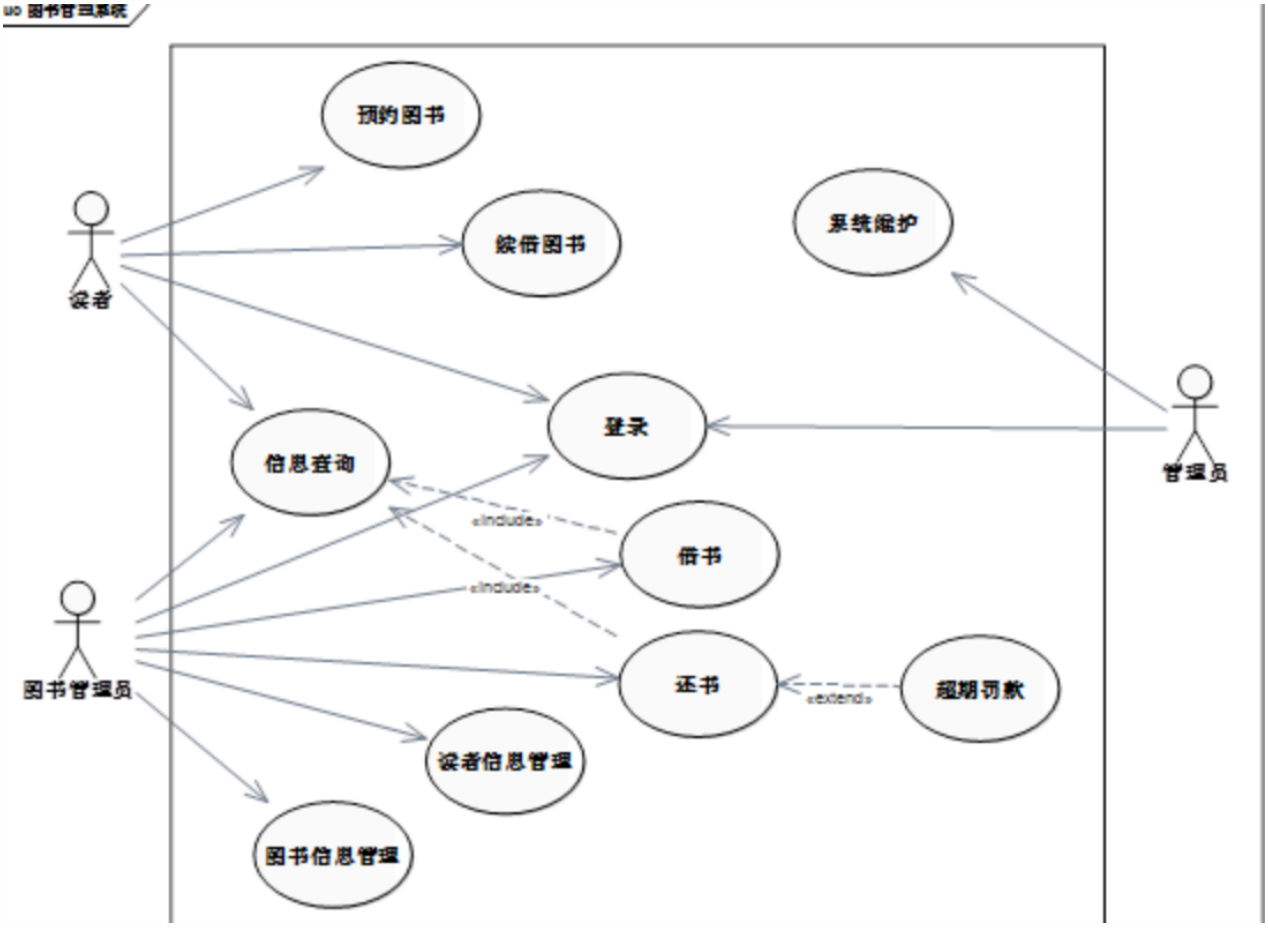
用例图是指由参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图。

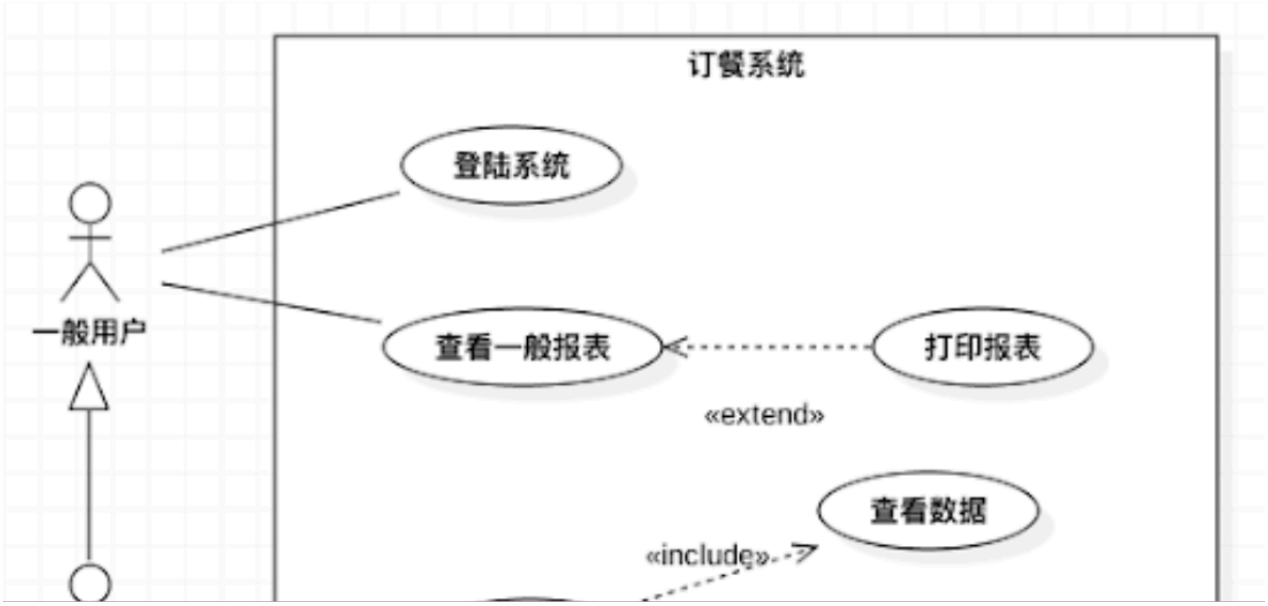
用例图（User Case）是外部用户（被称为参与者）所能观察到的系统功能的模型图。用例图是系统的蓝图。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。

其中有几个概念:

1. 参与者: 不是特指人，是指系统以外的，在使用系统或与系统交互中所扮演的角色。
2. 用例：是对包括变量在内的一组动作序列的描述，系统执行这些动作，并产生传递特定参与者的价值的可观察结果。
3. 系统边界: 是用来表示正在建模系统的边界。边界内表示系统的组成部分，边界外表示系统外部。
4. 箭头: 用来表示参与者和系统通过相互发送信号或消息进行交互的关联关系。
5. 作用:（1）获取需求；（2）指导测试；（3）还可在整个过程中的其它工作流起到指导作用。

我从网上找了几个用例图，其中的每一个圆圈，都代表一个功能。


uo 图书管理系统
预约图书
续借图书
系统维护
读者
登录
信息查询
管理员
«include»
借书
«include»
图书管理员
还书
«extend»
超期罚款
读者信息管理
图书信息管理



通过用例图，可以一目了然的看到系统提供的所有功能。

5. 写用例描述

但是用例图没有详细的描述每一个用例的执行过程，也就是说：用例图从总体上描述了系统的需求，但是没有描述行为过程。

因此，我们可以对每一个用例，附上一个简单或详细的用例描述，这样就更加具体的确认了这个用例的行为过程。

下面也是从网络上找的 2 个用例描述示例，可以看到并没有统一的格式，需要根据项目的性质进行增减。

用例描述	用户对指定文章进行本地收藏	
参与者	用户	
入口	目的地攻略文章卡片页或文章详情	
前置条件		
后置条件		
基本事件流 1	用户	系统
	1.用户点击收藏按钮（未收藏状态）	
		2.将该文章归类至收藏标签中，并高亮按钮；
		3.显示文字弹出框“收藏成功”，两秒自动消失；
备选事件流 1	用户	系统
	1.用户点击收藏按钮（已收藏状态）	
		2.将该文章从收藏标签中移除；并取消按钮高亮；
		3.显示弹出文字“已取消”，两秒自动消失；
业务规则	1.收藏文章的数据为本地数据，无帐号同步； 2.被收藏数量记录在运营系统中；用户删除应用再重新安装的收藏数量叠加； 3.已收藏的文章被禁用或失效后自动删除	
备注		
涉及业务实体		
影响平台		

编号	<用例编号>	名称	<用例名称，与用例图一致>
参与者	<参与者角色>	优先级	<需求开发优先级>
概要描述	<说明此功能的业务使用场景或具体目的>		
前置条件	<用例执行前系统和参与者处于的状态，是用例开始的必要条件；例如权限条件>		
用例描述			
流程名称	<主流程1 流程名称>		
触发事件	<此用例的触发交互，例如点击按钮等>		
步骤	参与者	系统	
1			
2			
3			
流程名称	<分支流程1-1 用例主流程外的分支，例如登录系统过程中取消登录>		
2			
3			
流程名称	<异常流程1-1 用例过程的报错，例如密码错误、超出限制条件>		
3			
结束输出	<用例主流程完成后的输出，例如进入系统首页>		
后置条件	<用例执行后的限制条件，例如最后一页无法点击下一页>		
界面描述			
UI示意图	<Demo截图>		
列表	<名称、排序、说明（规则 公式 限制）>		
按钮	<名称、说明（交互说明，例如点击、双击、置灰）>		
其他界面元素	<输入框、选择项、时间样式、弹窗样式等>		
备注	<其余补充说明、性能等>		

6. 确定关键需求

假设我们在不断的搜集和分析中，尽可能地列出了所有的需求(功能需求、质量属性、条件约束)，下一步需要做什么呢？需求这么多，该从哪一个需求入手呢？

关键需求 = 关键功能 + 关键质量。它确定了架构的大方向。

首先要明确一点：不可能所有需求都是一律平等。我们要从众多的用例出找出下面这 3 类需求：

1. 关键功能需求：那些涉及到的模块最多、模块之间协作方式最有代表性的功能，筛选出关键功能子集；
2. 关键质量属性：在开发和运行阶段，哪些质量属性对软件架构的影响比较大，如果质量属性之间存在相互矛盾的地方，应该优先考虑哪个？
3. 高风险部分：从技术难度上考虑，哪些功能在技术实现上存在风险，需要体验提前进行技术验证？

这 3 类需求就是需要我们重点照顾的需求，也是进行下一个步骤(领域建模)的输入材料。

由于篇幅太长了，关于设计部分的内容(领域建模、概要设计和详细设计)，下一篇文章我们再继续，希望能对你有所帮助！

好文章，要转发；越分享，越幸运！

推荐阅读

【C 语言】

C语言指针-从底层原理到花式技巧，用图文和代码帮你讲解透彻
原来gdb的底层调试原理这么简单
一步步分析-如何用C实现面向对象编程
提高代码逼格的利器：宏定义-从入门到放弃
利用C语言中的setjmp和longjmp，来实现异常捕获和协程

【应用程序设计】

物联网网关开发：基于MQTT消息总线的设计过程(上)
物联网网关开发：基于MQTT消息总线的设计过程(下)
我最喜欢的进程之间通信方式-消息总线

【物联网】

关于加密、证书的那些事
深入LUA脚本语言，让你彻底明白调试原理

【胡说八道】

以我失败的职业经历：给初入职场的技术人员几个小建议