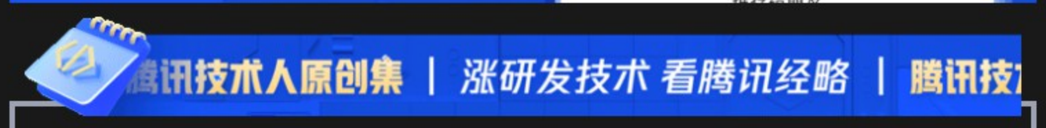
架构师必备底层逻辑：设计与建模

原创 boris 腾讯云开发者 2024年08月07日 08:45 北京



👉目录

程序员往往习惯于接到需求立马开始撸代码，原因无非是需求急任务重老板盯得紧。但在实际的开发场景中，我们往往会发现，写完代码，需求变了；人力多了，质量差了；业务代码，写起来没劲……

在追崇多人协作的现代软件开发体系下，这些问题背后的前置解决方案，其实就是设计和建模。本文将带你深入软件开发的初始，了解写代码前要做的几件事。

01

软件开发需求现状与实际困局

古早时期的软件开发，老夫写代码就是一把梭，Ctr C+V，单测是什么？直接上线！有问题下个版本再改，业务需求不等人，哪有时间做排期。

而随着互联网的不断发展，当前时代下的软件开发已经形成了集团军作战、体系化排需、工程化落地、自动化测试的正规路子。——从需求，编码实现，测试，发布的流程中不断优化，利用 CICD，加速迭代。

即便是业界广受推崇的“Two Pizza Team”，除了 2-3 个研发人员，也还会配备一定规模的产品经理，更别提测试、运维、运营等人员了。需求越来越复杂，人力堆砌就成了必然，想上线？先取号排期吧！

然而，研发流程确实规范化了，速度也确实提上去了，需求变更却还是让开发人员头痛不已，项目质量也在持续变差。老板还在“速度、质量、成本”的不可能三角中，一遍一遍地强调既、又、还……

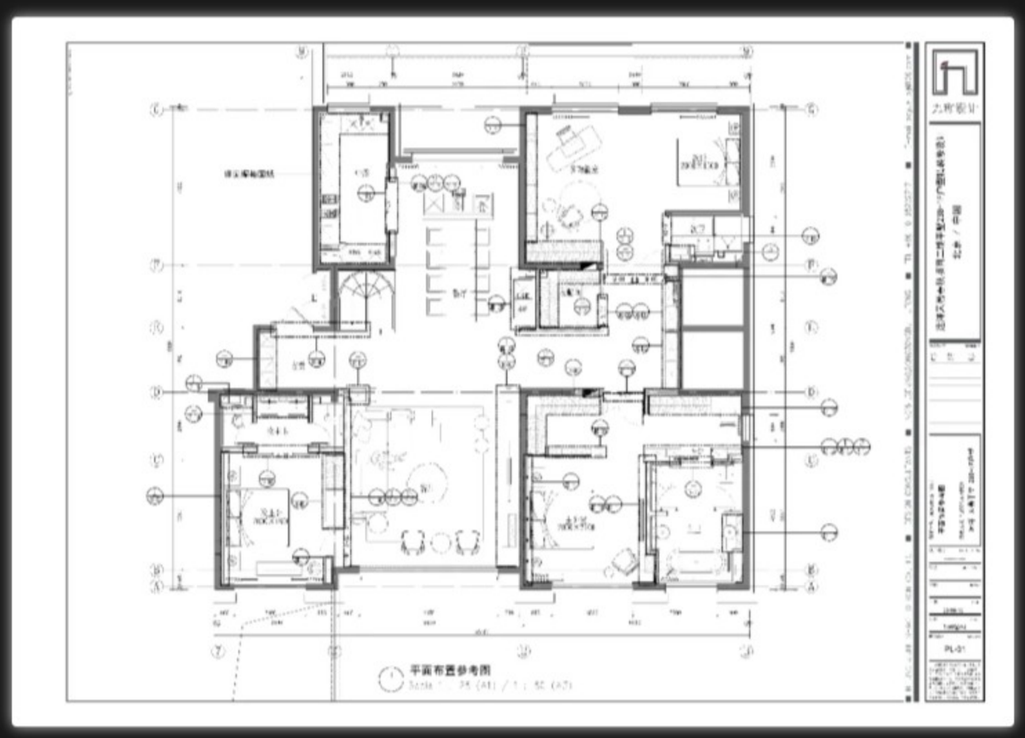
怎么办？结构性的问题往往无解，但动手写业务需求代码前，确实还有更多提高效率的工作可以做——设计和建模。

02

为什么要做设计和建模？

所谓架构师，是软件开发中那些对业务抽象做得最好的人，随着级别的提升，工程师所面对的需求会越来越抽象。承接抽象需求，提供抽象架构是架构师走向卓越的必经之途。

这有点类似于建筑行业，一栋摩天大楼的落地建成，往往需要特别精密的前期测算与图纸设计，在分析设计之后，按照图纸规划施工——写代码也应当如此。



设计建模的有效性源于，一，重新回到业务的跑道，跟业务一致；二，设计建模才能让协作真实有效；

研发和业务需求的摩擦，本质是研发实现跟实际需求不一致，无论是研发走偏了，没有理解需求，还是需求本身不能满足涉众的利益，都会使得最终上线的功能需要回炉重造，折磨项目组的成员。从设计的语言上看，设计的层次有所不同，不仅仅有代码细节上的设计，也有业务上高层次的设计，高层次的设计是用业务的术语去表达，最贴近业务实际情况，也能帮助研发同学发现业务中不合理的点。

进行设计建模能够让协作变得有效，一方面，设计建模前期是沟通和信息对齐，将协作的内容提前，一方面，采用合适的图形化工具，review 的成本是相对较低的。

03

设计和建模的三个关键点

3.1 业务建模

业务建模主要聚焦于分析涉众利益，厘清业务流程。从工具上来说，主要是用例图，流程图；从内容上来说，主要是找人（利益涉众，系统执行者），找业务实体（其余系统，相关的重要对象）。

分析涉众利益：

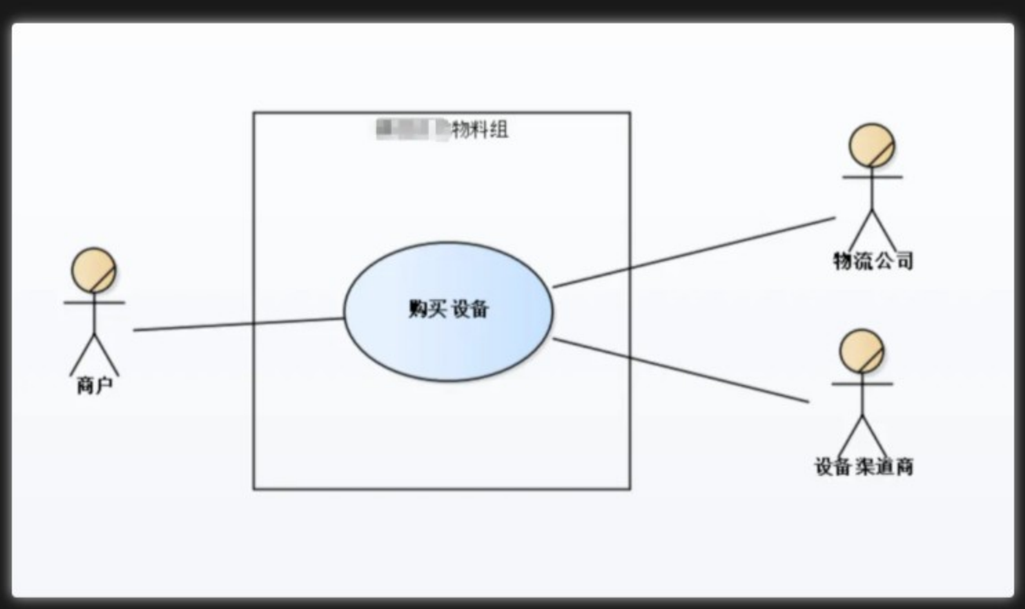
1. 找到软件产品的愿景，愿景表达了软件产品带来的核心意义。
2. 找到利益相关的的涉众和其利益诉求。

一部戏演什么内容，不是演员决定，而是由台下各种观众的口味角逐而定。观众按照重要性排排坐，优先照顾第一排的口味 、然后第二排、第三排.....同理，软件系统也要依次照顾各排涉众的利益。涉众利益之间的冲突和平衡，决定了系统的需求。对于实在照顾不到的后排，很多时候只好抱歉了，这个系统可能会损害你的利益。

业务用例图：

知道了涉众的利益之后，就要分析业务流程，并对现有的流程进行改进。软件产品没诞生之前，业务是如何被处理的，找到原来业务的处理方式则可以梳理出业务用例。

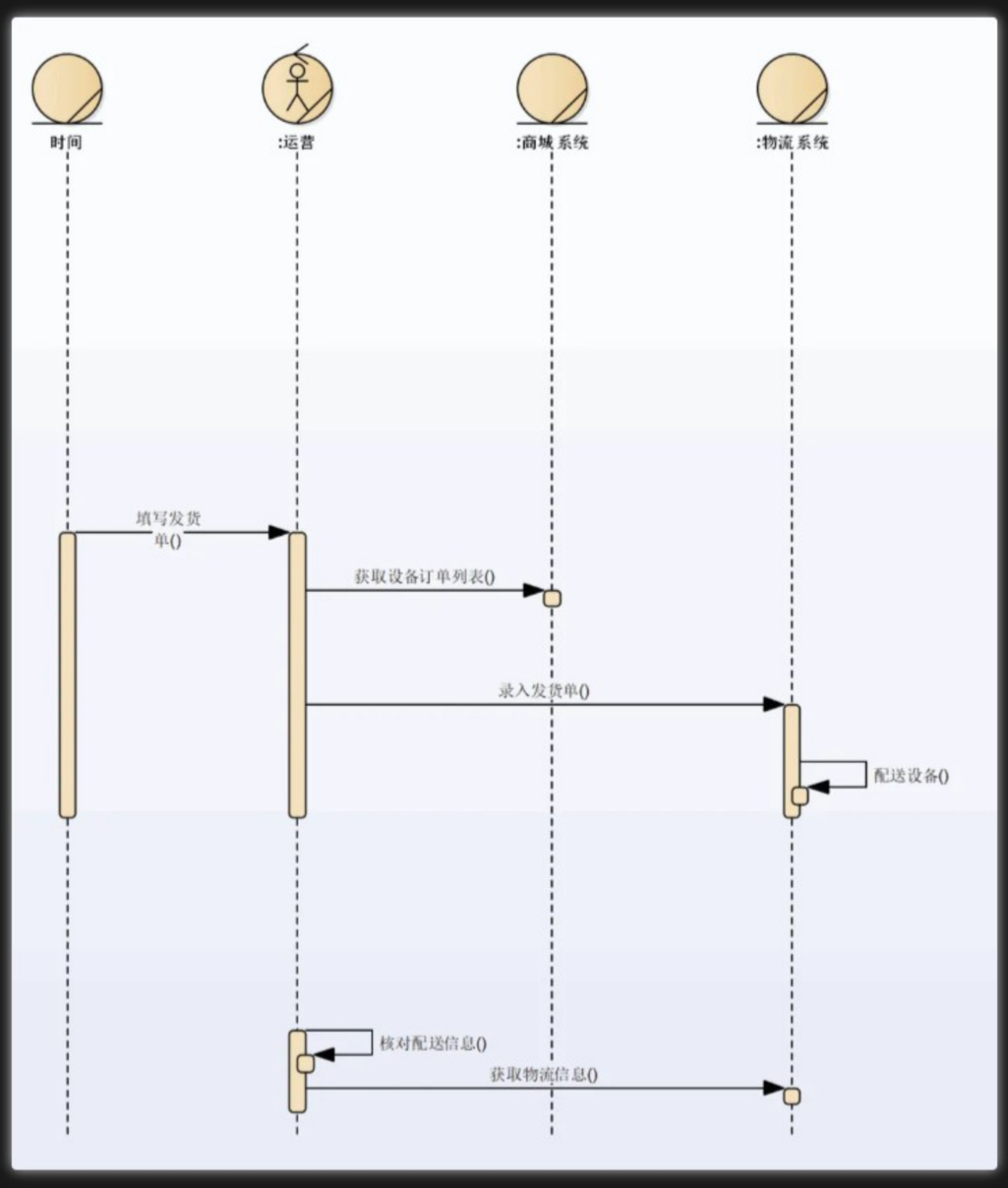
以一个商城购买配送系统为例：



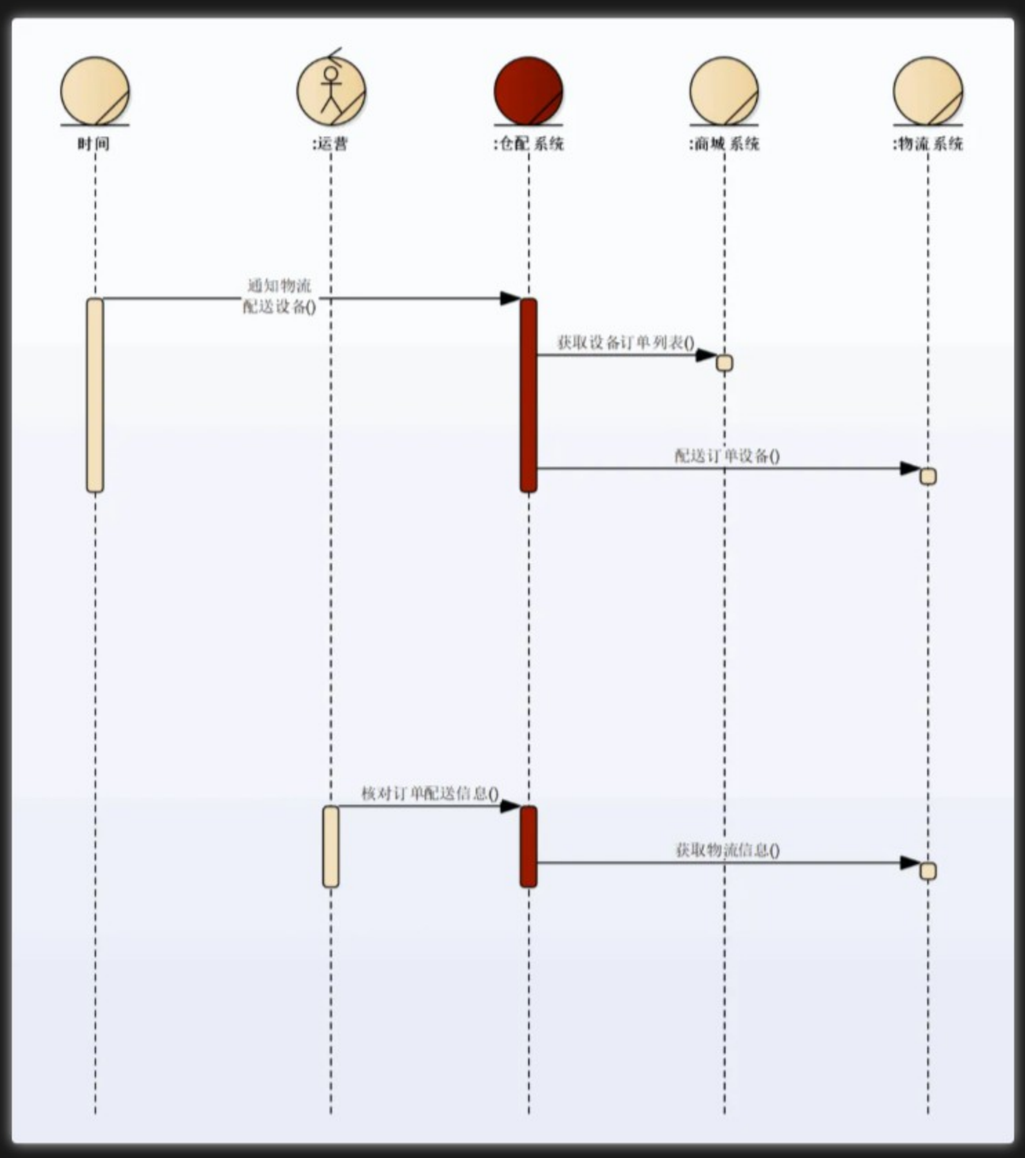
物流公司和设备渠道商都是辅助购买设备的执行者，因此放到右边。值得注意的是，业务用例要体现价值，虽然在实际业务流程中，商户同时做了很多事情，比如签收设备，但签收设备不能反映业务价值，故而只有一个购买设备的用例。

3.2 业务流程分析

完整的业务流程图可能很庞大，需要关注的是其中最有可能影响涉众利益的流程片段，如下为购买设备业务流程中配送设备的流程片段：



这里的业务流程问题体现在，配送设备的流程是在全部业务流程中较为繁重，人肉工作量大的流程片段，不符合涉众利益。所以系统需要改进流程，替代人的部分工作，如下图。



业务序列图中，每一个箭头代表的是职责，在业务序列图中，需要考虑的是职责的层次问题，过小的职责放入流程图中，会导致信息过载，忽略最有价值的职责。

业务流程分析是一件很复杂的事情，研发同学可以利用需求中的信息，同时加上自己跟涉众的日常沟通和调研，把握核心的涉众利益，业务用例和业务流程，就可以解决大部分在需求上的理解偏差问题。

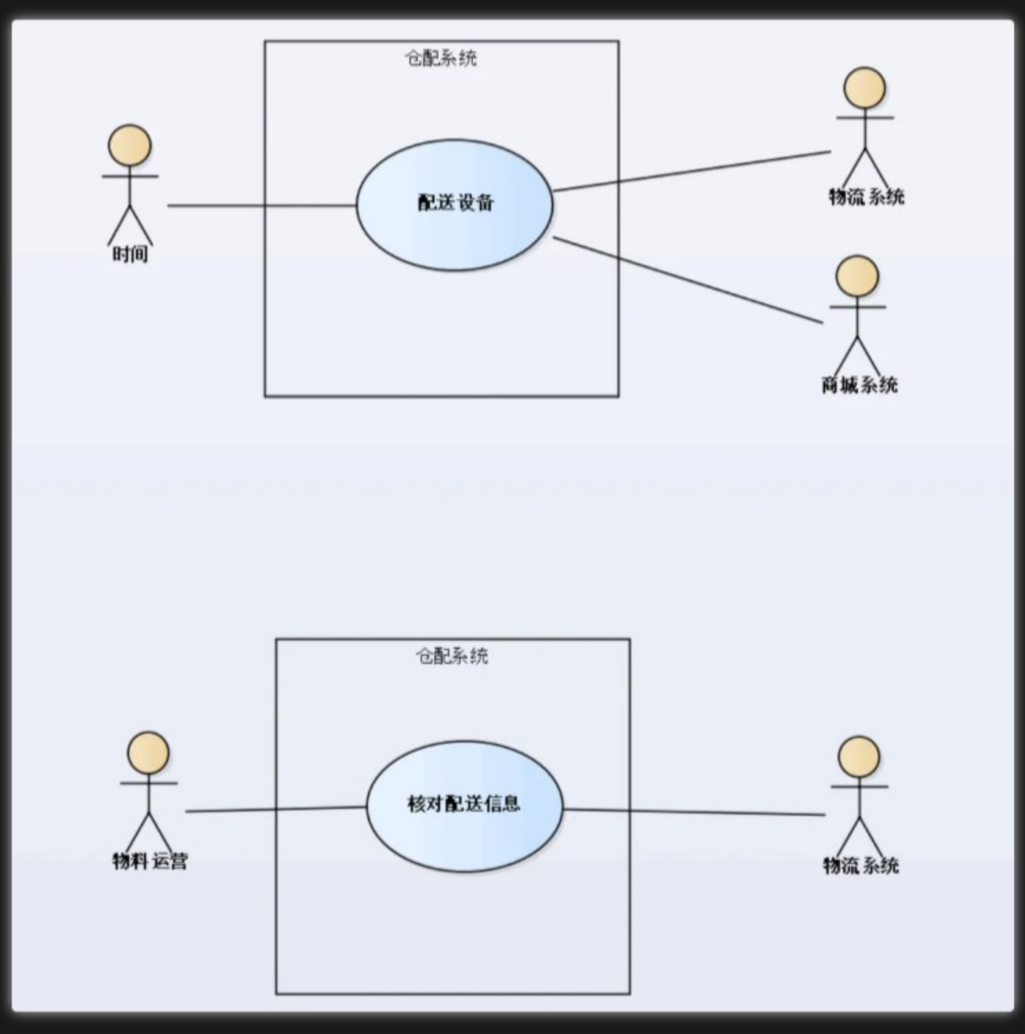
3.3 系统建模

系统建模关注的是系统与外部的边界和系统自身的职责，主要聚焦两件事：1，画出系统用例；2，写出用例规约。

系统用例图

系统用例图是业务流程中，系统执行者与系统发生的有价值的交互。系统执行者可以是人，可以是外部系统，甚至可以是时间。系统用例要体现系统的价值，系统会做很多事情来实现业务价值，我们应当关注业务价值。有些是低层次的职责，没有体系具体价值，如："获取商城订单信息"是为了配送订单中的设备而发生，应当关注"配送设备"。

如下是仓配系统的系统用例图：



系统用例规约

有了业务流程图和系统用例图，需要根据进一步细化系统边界上的约束，保证系统的稳定性。系统执行者与系统的交互细化了详细的约束，系统的稳定性才能提高，如果没有仔细列出约束，有可能会忽略一些边界条件，导致系统的故障；如：仓配系统不考虑来自第三方商城订单要配送的设备数量限制，则会因为第三方商城出现的错误，导致资产损失。

系统约束来源于系统用例，根据业务的规则，详细描述业务流程中的基本路径，扩展路径和约束。值得一提的是，约束不是告诉研发同学如何实现功能，约束是指业务规则，厘清系统执行者与系统之间的边界，以及边界上的约束。设计评审的时候，可以关注关键路径上的安全规则是否到位，这么做对提高系统的安全稳定有极大的帮助。

3.4 类的分析与设计

识别类

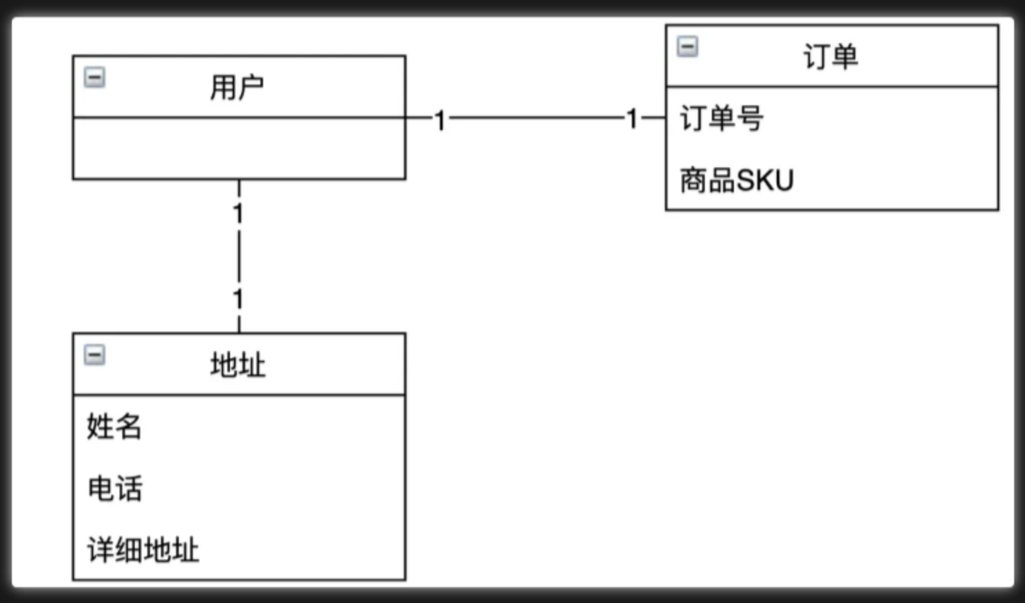
常见的类一般有三种——边界类、控制类和实体类。边界类是外部系统在系统内部的映射，借由边界类，系统和外部系统交互。一些接口请求、输入输出都属于边界类的职责。控制类往往体现用例流程，一般情况下，一个用例就是一个控制类。实体类是系统的核心，良好的实体类设计能够提高系统的复用程度，减低系统的复杂性。

找实体名词

识别具体的类需要去找业务流程、系统流程、系统规约中经常出现的名词。在前文的流程图中，订单、商城、设备、物流、用户是反复出现的名称，说明这就是类的业务实体体现。

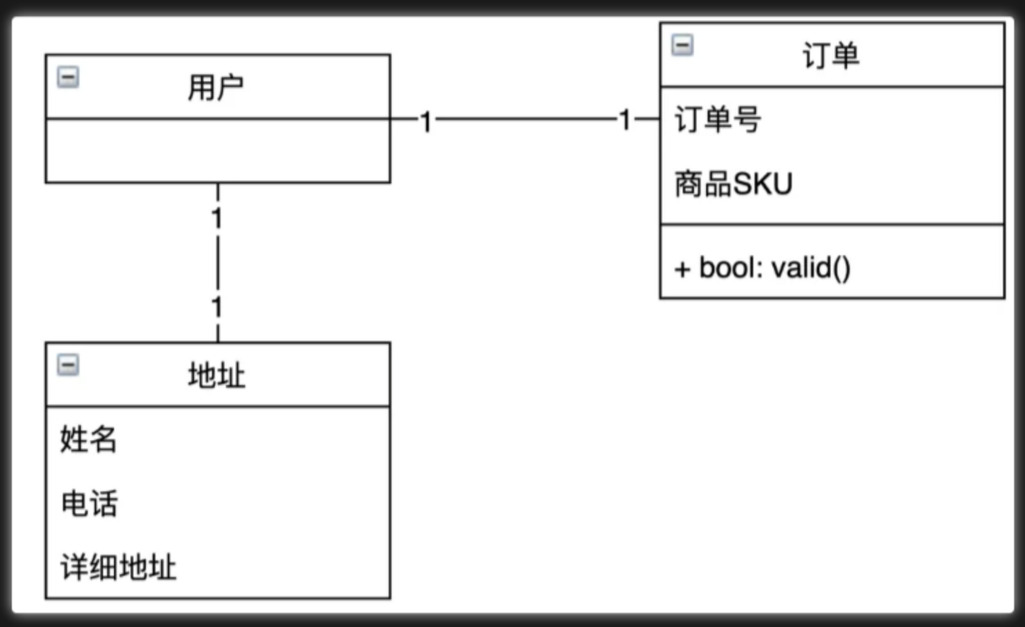
找属性

类的属性不是凭空产生，需要对业务实现有价值。找到那些对于系统实现必不可少的属性，放到正确的类中。如仓配系统中的订单，包含订单号，商品，用户。用户则有收件地址。



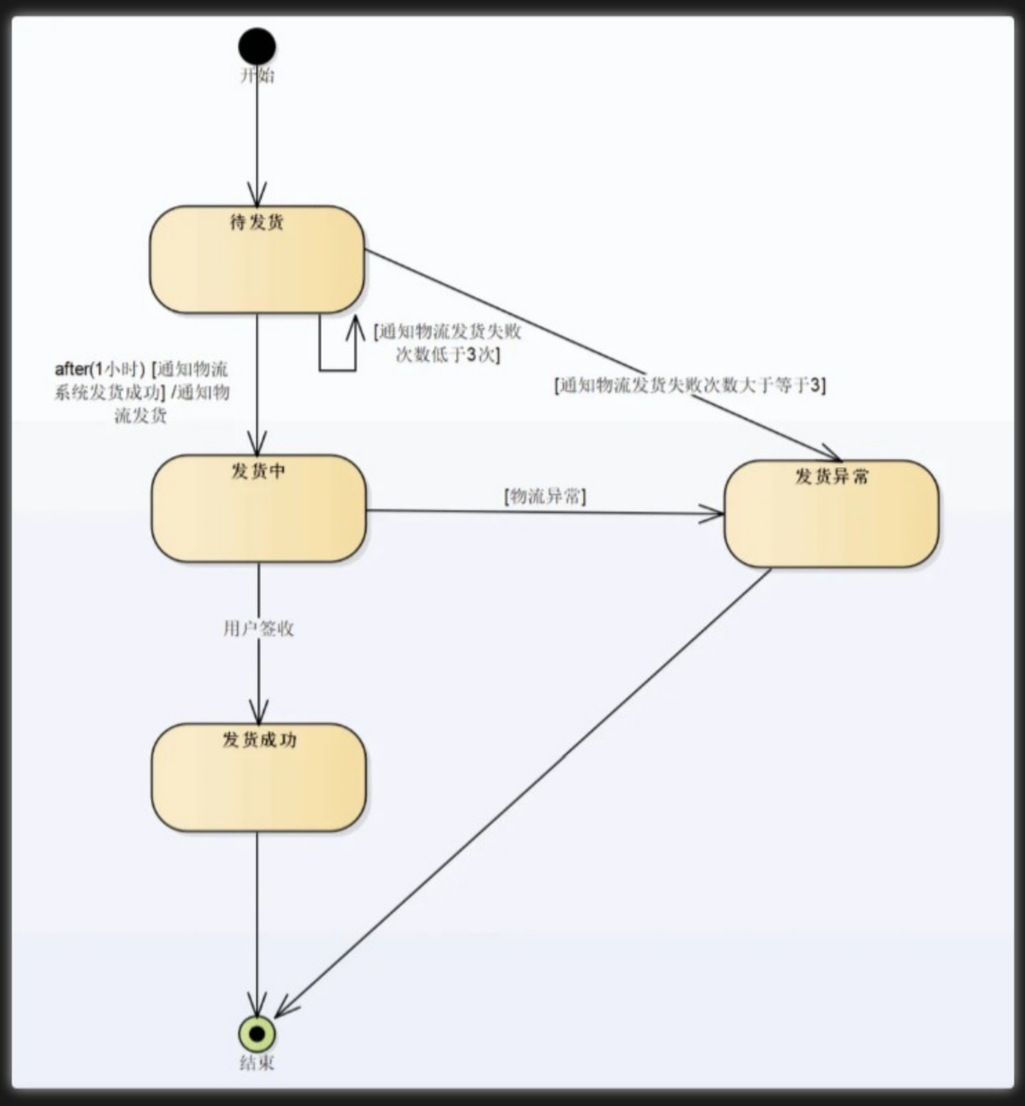
找职责

业务规则和约束中，可以找到一些实体应当有的职责。以订单为例，其就具备验证合法性的职责。



状态机

对于一些主要的实体类，还需要设计出他的状态机，清晰的状态机能有效地厘清系统内的一些事件和状态，增强系统整体的健壮性。



以仓配系统中的订单为例，从用户购买的待发货状态，到通知物流发货，再到实际发货，物流签收有一系列状态的演变，这都要体现在状态机上。

04

总结

不同的建模理论中提到的名词概念很多，每个概念单独拿出来讲都可以写一个章节或者写成一本书。建议一开始不要陷入各种复杂概念的理解和纠结上，先去充分地理解何为面向对象，建模的本质是映射，再去回看各种建模书中阐述再多的道法术，因为这些招式都是围绕寻找软件对象世界中的人，事，物和规则展开的。这里也非常推荐大家去阅读《软件方法》、《软件建模与设计》等经典书籍。

写好代码只是软件实现的工程手段，如果没有好的设计、建模，再好的代码可能也是一种浪费。你说呢？

-End-

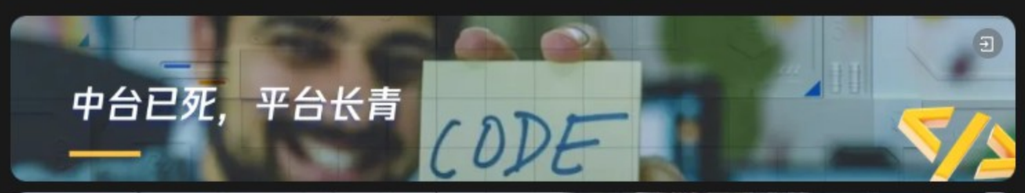
原创作者 | boris

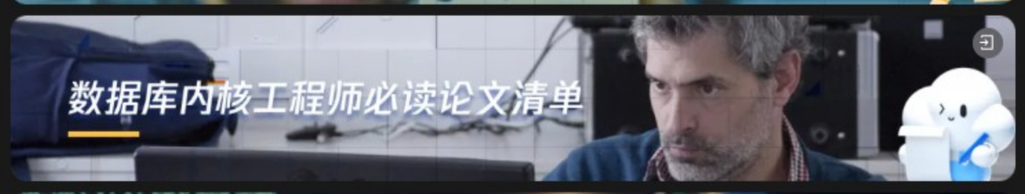
你觉得怎么样才算是好的设计与建模呢？欢迎评论分享。我们将选取**点赞本文并且留言评论**的一位读者，送出腾讯云开发者定制发财按键1个（见下图）。8月15日中午12点开奖。

欢迎加入腾讯云开发者社群，享前沿资讯、大咖干货，找兴趣搭子，交同城好友，更有鹅厂招聘机会、限量周边好礼等你来~

（长按图片立即扫码）

「 精品·知识推荐 」





腾讯云开发者
腾讯云官方社区公众号，汇聚技术开发者群体，分享技术干货，打造技术影响力交流社区。
925篇原创内容

公众号

赛博玄学，一键三连少一个Bug!

为好文章 点 收藏 点亮

腾讯技术人原创集 234 # 架构师 5

腾讯技术人原创集 · 目录

‹ 上一篇
服务端开发必备：9大性能优化秘技

下一篇 ›
9本醍醐灌顶的计算机好书