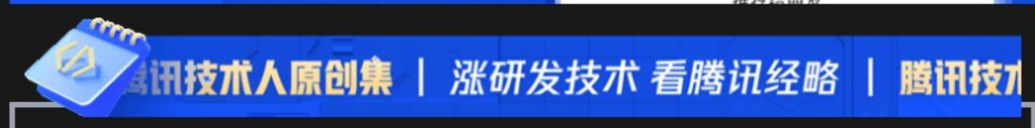
架构师基本功：如何画好一张UML用例图？

原创 欧霄 腾讯云开发者 2024年08月14日 09:36 北京

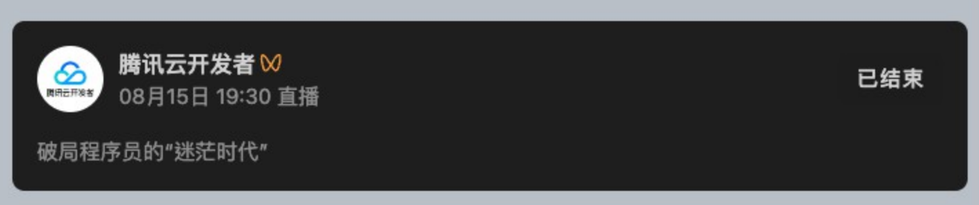


👉目录

在做程序设计的时候，开发同学往往都有类似的困惑：分不清楚业务用例图、系统用例图都是什么，二者的区别是什么，也不确定自己画的图对不对，会不会被评审挑战。

本文作者从业务建模角度切入，详细拆解了业务用例图和系统用例图的分析与画法，相信能够帮助你在之后的设计中画图信手拈来、得心应手。

号外！「鹅厂程序员面对面」本周四晚直播精彩继续，预约观看有机会抢鹅厂周边好礼！



01

从业务建模到业务用例图

1.1 愿景

做任何事情，第一步是要想清楚为什么去做，有没有价值，给谁带来价值，如果这些都没有想清楚，那事情多半走不了多远。

而业务建模的第一步，也是先要理清楚我们的愿景是什么。

> 如果缺乏清晰、共享的愿景，开发人员会在错误的方向上狂奔，做得越多，浪费越多，却反而还乐在其中。
>
> 《软件方法》

考虑愿景时，我们应该问自己的问题是，东西最应该卖给谁，以及对他有什么好处。

思维不能停留在做一个"可以工作的软件"，而应是"可以卖的软件"，这两者是有很大区别的。

同时，愿景也应该与团队内所有人共享，只有大家都有清晰的愿景，才能把力往一处使。

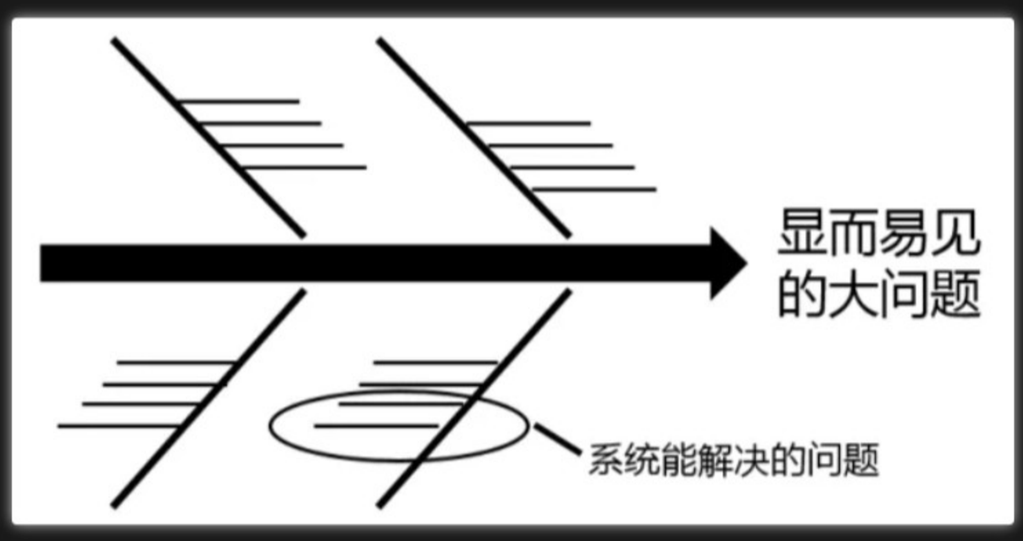
那如何得出清晰的愿景呢？

1. 定位目标组织及其"老大"（也就是系统最优先照顾其利益的那个人）
    a. 定位人群：人群属性必须细化再细化，要找到最有代表性的老大进行调研，而不是大街上随便找一个人，随着互联网发展越来越完善，想要得到市场，只有深耕行业，深耕的意思就是要细化人群，好比要做一个文本编辑器，Word、WPS 等已经很好地满足了绝大多数人的需求，你再去做一个标准编辑器很难卖出去，更好的选择是选择一个细分领域，如医生的病历编辑器，然后把调研的人群定位于医生群体；

b. 定位机构：先明确机构的范围、要替换的既有系统（人脑或电脑），老大应该是目标机构的主管或负责人，且是业务负责人，而不是 IT 负责人，也不是企业最大的领导。在和传统企业或机构的合作中，我们往往会对接一个信息中心主任，但要明确老大应该是最了解业务的人，而不是管信息化的人，只有这样才能靠近最真实的需求。

2. 提炼改进目标。要注意的是，目标应该是改善组织行为的某个指标，如 IoT 设备的材料提交速度、审批步骤等，而不是系统的功能，也不是系统本身的规模、质量，这些都是基于指标分析出的需求和结果

愿景必须有具体的定义，不能是“正确而无用的废话”，大而泛的目标谁都会提，真正具有指导意义的，是具体的细节定义。



业务示例：

假设我们想打造一个 IoT 管控业务中的设备售后咨询系统，那么我们首先也需要明确我们的愿景。

如前所说，愿景需要明确人群机构，还需要有明确清晰的目标定义。那么一个 IoT 管控业务的售后系统愿景大致如下所示：

1.2 业务用例图

当我们有了愿景，就等于明确的知道了“老大”及其代表的组织对哪个现状指标不满意，而这也是我们要做的软件系统的目标。

然而知道目标不代表知道怎么达成目标，要解决组织存在的问题，就要先分析组织的现状。

分析现状实际上就是分析我们所要涉及的领域，其业务是如何流转的，然后基于现状，再去寻找其中的改进点，以达成我们的愿景目标。

这里对业务现状的分析，就是以业务用例图的形式进行。

业务的用例图，表示了从外部看，一个组织的价值。

业务用例图有几个组成部分：

- 业务执行者：指组织的边界外，和组织交互的其他组织（人群或机构）；
  - 业务的执行者应该是组织，而不是个人或者系统（因为他们都可能被替代），只有组织才反应了业务的交互；
  - 时间不应该被作为执行者，而应该是对应的业务组织。
- 业务用例：业务执行者希望通过和所研究组织交互获得的价值（如取款、存款等）。它代表了组织的本质价值，很难变化，会变化的只是用例的实现——业务流程（如取款机取款流程、网银转账流程等）。

业务流程内，才对应具体的业务对象：

- 业务工人：组织内部的人，可被其他业务工人或业务实体替换；
- 业务实体：组织中的非人智能系统。

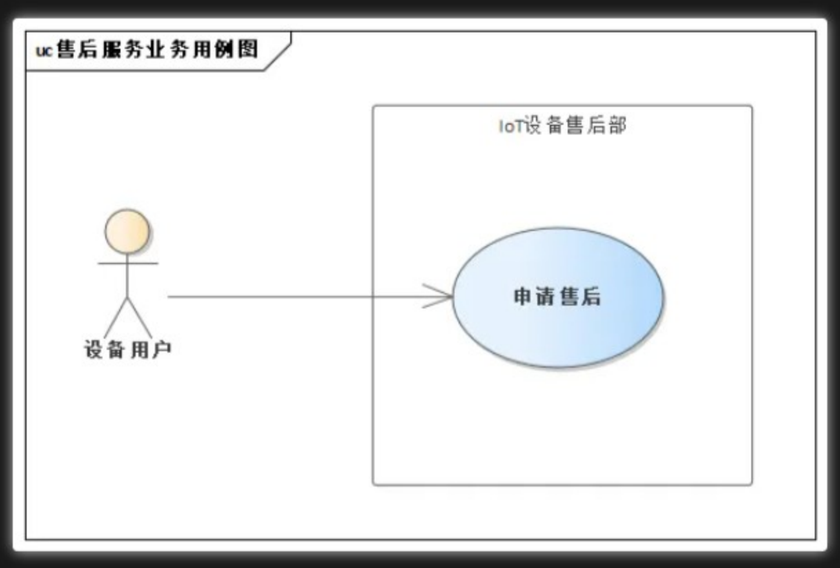
但业务工人和业务实体不需要在业务用例图出现，他们不是组织的价值，而是成本，时刻记住，业务用例图表现的是组织对外部的价值。

所以业务用例图往往很抽象，很简单，在这一步，我们的目的是理清我们所研究的组织的价值所在。

而我们要做的，是对这种有价值的业务用例，进一步研究其业务流程，找到改进点，而这就是我们软件系统的着手点。

业务示例：

对于我们的 IoT 管控售后系统来说，业务用例就是我们设备的用户来咨询运营问题，并由运营人员解答问题，这个就是售后部门的价值所在，如图所示：



1.3 业务序列图

在找到业务用例后，我们需要进一步研究用例的业务流程，这需要借助业务序列图的帮助。业务序列图描绘的是该业务用例的流程**现状**。

要如实地画出**现状**的序列图，只有描绘现状，才能找到改善的途径。

而不能是描绘想象中的或打算做的样子（这会导致改进点可能根本不合适或者已经被实现了）；

也不能是组织给出的所谓规范（那是理想情况，实际流程可能与规范大不相同，毕竟人都会钻空子或者节省力气，如何保障流程按规范进行，也是一种改进点）；

更不能因为是创新产品就认为没有现状（大部分创新都是基于某个既有流程做的改进，很少有凭空创造需求并成功的）。

业务序列图主要由业务对象和消息构成，长得很像研发人员熟悉的时序图。

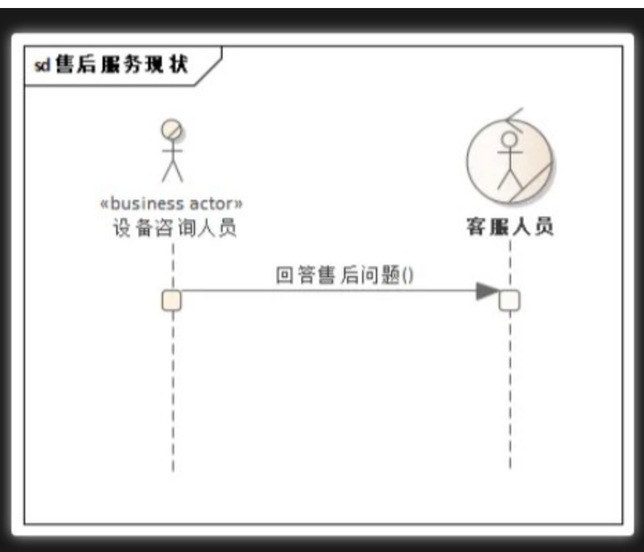
序列图上的业务对象的最小颗粒是人和非人系统；消息是指业务对象 A 请求业务对象B做某事，或A调用B做某事的服务，做某事是 B 的责任。

在画业务序列图时，要注意以下几个反模式（不好的做法）：

- 业务对象突然细化到某个存储载体，如“关系型数据表”，这不是改进业务流程所关心的（除非改进的是一个数据库系统之类）；
- 业务对象扩大到某个组织，而不区分内部的人或系统，这会导致可能遗漏改进点；
- 关心与核心域流程不影响的各种系统（如通讯工具、word 文档等），这同样不是业务流程关心的（除非你就是要改进这一点）；
- 把定时器当做执行者（业务对象），执行者是时间，定时器只是和时间打交道的边界类；
- 赋予业务对象超出其能力的责任，如发送写专利的消息给 word，word 本身并不会写专利，专利是专利人写的，word 只用于“编辑专利文档”；
- 包含系统内的多次细致的交互流程，过于繁琐，又不在改进点内；
- 把一个不具备智能的消息传递参数（如某个物品）当做业务对象；
- 消息内容中包含“请求”二字，箭头本身就包含了请求的意思。

业务示例：

这里我们尝试绘制一下 IoT 管控的售后服务现状的业务序列图，经过了解，业务的现状是，使用设备的用户会直接咨询运营人员，提出他们使用过程中的疑问，而运营人员则需要对疑问一一作出解答：



画好业务序列图后，我们就要看看这个图中，存在什么改进点，可以被我们的软件系统所解决，也就是有没有可以让我们发挥价值的地方。改进模式一般分以下几类：

- 把物流变成信息流（物体和人的流动，改为信息的流动，常见于各种信息化、线上化）；
- 改善信息流转（减少人与多个系统打交道的流程，减少系统间沟通不畅）；
- 封装领域逻辑（把人的经验与思考封装到系统中，需要深耕领域业务，现在这种改进的比重越来越大）。

业务示例：

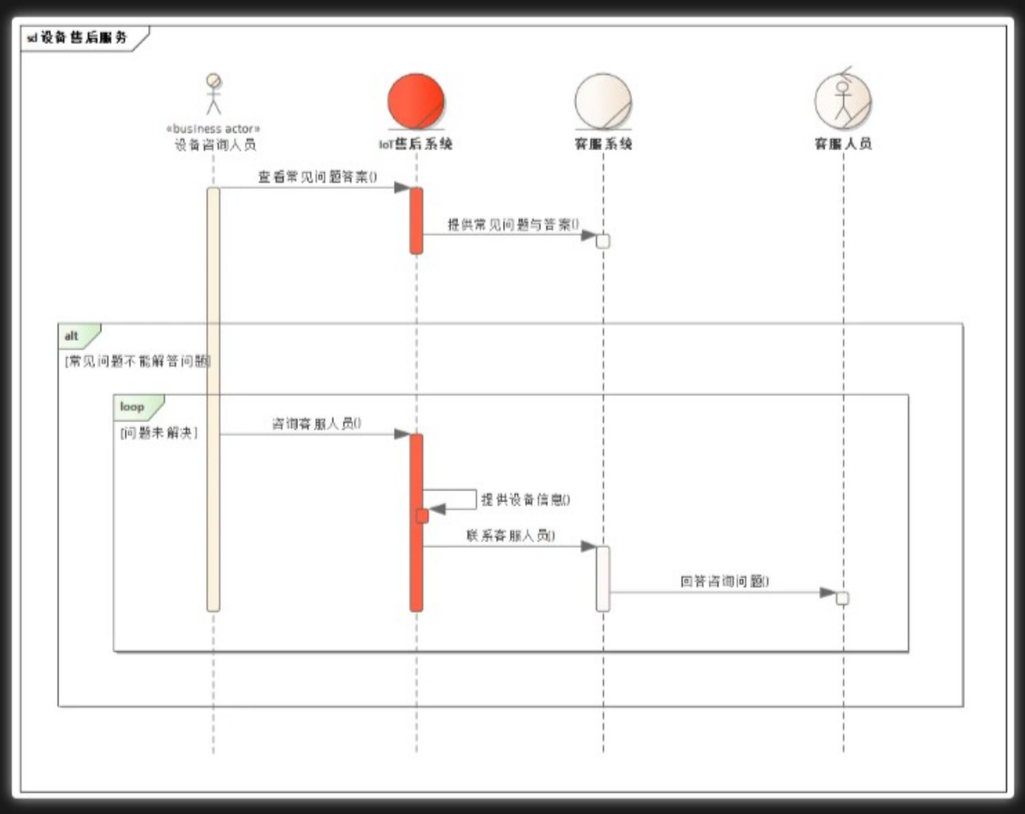
在 IoT 管控的售后服务现状序列图中，我们可以发现，售后服务的现状全部都由人力组成，其问题和解答的流动是由人来传递的。

那么改进点就很好找了，我们尝试“把物流变成信息流”，以达到我们的愿景，即：在不影响用户售后咨询体验的前提下，减少客服人员处理售后问题的工作量。

我们在这个业务流程中，引入客服系统。

一方面沉淀常见问题，供用户快速获取答案。另一方面让客服来回答一些比较常规的问题，解决用户的进一步人工咨询的需求，把我们的运营人员从重复的咨询解答劳动中释放出来。

此外，我们也搭建一个售后系统，作为连接用户与客服的桥梁，实际上这个售后系统包括微信小程序、后台模块等，但作为研究过程来说，他就是一个整体的售后系统。那么改进后的业务序列图便如下所示：



如果发现一个业务流程已经很完善了，很难找到常规的改进点了，怎么办呢 ？这里有一些建议：

- 遇到困难、资源受限等问题，展开想象，打破思维障碍，而不是接受现实做一个普普通通的系统；
- 用廉价的方案来“山寨”富豪、高官的生活（如下属们周到的服务安排），再把这种生活复制给普通人；
- 观察和调研领域内最成功的组织，找到其中的借鉴点，不能闭门造车。

— 02 —

从需求设计到系统用例图

当我们列清楚了领域业务的现状，也想清楚了应该怎么改进来体现我们的价值的时候，就可以开始研究我们的系统了。

实际上，就是我们要开发一个系统，需要列出有哪些需求点。

我们系统要实现哪些流程？这些流程有什么约束条件和步骤？怎么找到涉众真正的需求？这些都是我们在进行系统设计前要明确的，有的才能放矢。

2.1 需求启发

如何找到改进点，上文已经有建议，这里更加深入介绍一下，当要具体考虑需求细节前，我们需要尽可能地多从涉众处了解到他们的真实需求，前文也说过，要找到真正的涉众和“老大”，去得到需求的启发。在和涉众沟通时，应该尽量以其能够快速理解的方式，而不是画一个UML 图去沟通。UML 图适合在软件开发团队内部作为专业的交流手段，和涉众交流时，则应该视其习惯转换成合适的形式来获取需求，他们熟悉网页界面，就拿原型沟通，他们熟悉文件材料，就拿文件讨论。

做需求启发时，有几个手段和注意事项：

- 与涉众沟通前，要充分调研好资料，做好知识储备，才能让沟通过程有价值，不能什么也不清楚就去问，没人愿意花时间教你；
- 针对个体很多的涉众，可以问卷调查，但要注意避免无效答卷（可以埋钉子问题，筛选出明显乱答的）；
- 做访谈，和真正的不同涉众进行访谈，不能只找好交流的人交流（如只咨询熟悉手机的年轻人，不管中老年）；
- 善观察，观察涉众的环境、工作流程，甚至亲自去体验；
- 研究竞争对手，在竞争过程中，不同的阶段需要有不同的战略意识：
  - 开拓者：作为市场领先者，要开拓本领域，不要只关心追赶者；
  - 追赶者：研究领先者的优点，攻击其强势外表下背后的弱点；
  - 侧翼战：专攻细分市场做创新，也能占据一份市场；
  - 游击战：依靠地域优势、人际关系来生存，并逐渐凝聚出自己的特色。

通过需求启发，希望能够打造出尽可能有价值的系统来。

2.2 系统用例图

确定需求后，就可以画系统用例图。

我们之前曾经画过业务用例图，业务用例的重点是理出组织对外部的什么人群提供了什么价值。

而系统用例图就更细化一点，对外部的什么个体，提供了业务流程中的什么价值。

首先还是要明确“系统”这两个字。

什么是系统？系统封装了自身的数据和行为，能独立对外提供服务的东西。

注意，是能够独立对外提供服务，不是某个页面，也不是某个功能模块，更不是某个数据表。

系统的边界是责任的边界，而不是物理边界（如同一个系统可能包括不同服务器、移动终端等，但他们都是一个系统）。

系统用例图依然有系统执行者。

这里的系统执行者与业务用例图中的业务执行者也有所不同。

系统用例图中的系统执行者，是在所研究系统外，与该系统发生功能性交互的人（不是组织）或其他系统。

系统的执行者必须和系统有直接交互，不能是间接交互。

比如售票员是售票系统的执行者，但购票人不是。除非售票系统必须经过购票人的确认，那购票人就成了辅助执行者，但如果仅仅是展示信息给购票人看，那依然不是执行者（因为不需要等待购票人交互就可以完成用例）。

这里提到了执行者的类别，有两种：

- 主执行者：主动发起用例的交互，箭头从执行者指向用例。
- 辅执行者：在交互的过程中被动参加进来，箭投从用例指向执行者。

系统与系统执行者的交互必须是功能性的交互。

什么叫功能性交互呢，比如点击鼠标就不算功能性交互，鼠标也理所当然不应该是系统的执行者。

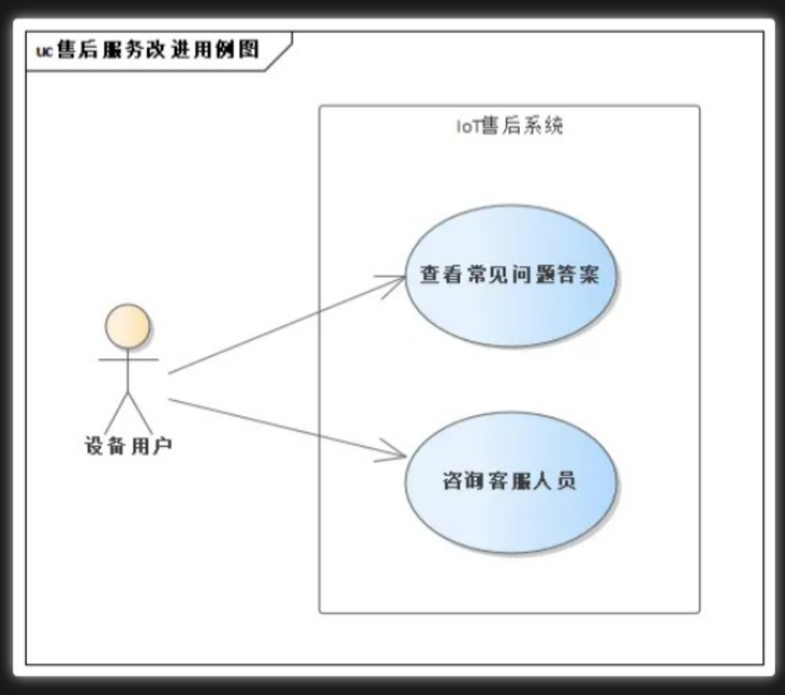
再来看用例，对系统用例图中的用例，也有以下几点要求：

- 在业务序列图中，从外部指向系统的消息，即可映射为系统的用例，所以画好业务序列图，也就能得到准确的系统用例；
- 用例必须是可以对执行者带来价值的，而不是任何一步繁琐的交互都算；
- 用例要明确其主要的目标客户，而不是谁可以来做就算作谁的用例，用例满足的是目标用户的期望；
- 用例不能描述为数据库某个表的增删改查，而应从涉众的业务需求出发，描绘真实使用场景；
- 不要把不同涉众的看起来实现类似的用例合并起来，比如把不同用户的查看行为合并为一个查看用例（如员工查看信息，组长审核信息），实际上他们的涉众、用途都是不同的：
  - 这里需要警惕：需求不要有“复用”的思想，如果考虑了“复用”，就要警惕是否转换到了设计的视角来思考问题；
  - 多个执行者不能指向同一个用例，如果真的无区别，那就应该泛化出更抽象的执行者，如果有区别，则应该区分为不同用例。
- 系统用例不用分层次，如果分了，可能是研究的对象没明确好；用例图中也不应该分模块、子系统，这是设计的思路；
- 用例命名，采用动宾结构，（状语）+动词+（定语）+名词。

业务示例：

在 IoT 管控的售后服务系统改进后的序列图中，我们已经把我们的系统进行了标红。

那么所有从外部指向 IoT 售后系统的消息，都可映射为我们系统的用例，这样系统用例图就比较简单：



2.3 用例规约

系统用例，描述了系统对外提供的一个个服务与价值，实际上是系统要达成的一个功能目标。

但要落实到需求方面，还远远不足，需要更加细化。

就好比我们日常接触到的需求单，只是一个标题明确做什么东西，是不够的，需要细化各种条件、约束、步骤、分支等等，用例规约就是在细化这些内容。

在画好用例图后，就可以进一步细化用例规约。

用例规约可以是文档的形式，但也可以直接画在 UML 图中，主要还是讲清楚细节。

一般来说，细节需要明确的会有下面这些内容：

1. 前置、后置条件：
   - 前置条件是用例开始前要满足的约束：必须是系统能检测到的！
   - 后置条件是用例成功后的约束，是某种状态，而不是一个动作。
2. 涉众利益：要考虑用例过程中，涉众的利益，涉众来自系统的人类执行者，要充分考虑上下游以及信息来源的涉众，综合考虑他们的利益诉求。
3. 基本路径：记录一个系统用例的步骤
   - 用例最成功和最核心价值的路径，就是基本路径；
   - 路径交互一般可分为四步：请求、验证、改变、回应；

- 路径书写的注意事项：
  - 时间的请求写“当到达时间周期时”；
  - 验证步骤不要写“是否”，直接写期望的结果；
  - 与辅执行者的交互可以是回应；
  - 主语要明确责任方（只能是主执行者或系统）；
  - 步骤要使用业务的核心术语描述，不要用不同的习惯用语；
  - 不要涉及界面交互的细节，这是设计约束；
  - 需求的判断标准是：“不这样不行”，而不是“这样也行”，这两者是有区别的。

4. 扩展路径：记录异常、意外情况处理的路径，需要注意：
   - 系统能感知到的意外才需要扩展路径；
   - 设计、开发不足导致的错误不是意外扩展，比如数据库保存失败，这不是需求需要关心的；
   - 不引起交互行为变化的选择分支不是扩展；
   - 界面跳转不是扩展。
5. 补充约束：其他可能需要明确的细节
   - 字段列表。
   - 业务规则。
   - 质量需求。
   - 设计约束。

有了以上这些信息，基本就能对需求有一个完整的描述和印象了。

03

总结

技术同学写的代码，只有落到业务上才能真正产生价值。企业的利润=需求 - 设计，真正带来利润、最为重要的，还是需求。把功夫下到这里，走对了方向，才能有所成就。

而分析与设计，更多地是如何完善自己的系统，如何做到快速响应细节的变化。

我们作为研发人员，在思考设计之外，也许更应该多想想，手头所做的事情到底有没有价值，我们的涉众，到底是真实的用户还是老板。

做真正有价值的事，真正投入去了解自己的业务，去理解接触的行业，其实就已经在另一个层面上，向更好的设计迈进了一大步。

如何画图只是实现思路具现化的一种手段，真正重要的是思考的过程。

面对一个新的领域、新的业务，或者新的场景，首先思考一下，你想要解决什么问题？想要带来什么好处？此乃愿景。

然后，面对领域/业务，想清楚他现在到底为涉众提供了什么样的核心服务，即业务用例。

接着，梳理好业务的现状，准确地理出目前的流程，得到业务序列图，至于怎么调研，方法很多，实践出真知。

再想一想，要达到你的愿景，可以对现有的流程进行什么改进？这所谓的改进，到底有没有给涉众带来价值？

如果业务场景还没有接入信息化，或者接入程度低，或许可以很明显地带来价值，但如果已经有其他的竞争者介入，你如何带来更大的价值呢？

这就是你要做的系统，而系统中涉及到的执行交互点，就是系统用例。

接下来就是我们最熟悉的东西了，想好需求，明确需求规约，细化出要实现的一切。

对于设计方面，也推荐一种相似的设计方法，叫做领域驱动设计（Domain Deiven Design，DDD）。

这种设计思想，就是通过边界划分，将复杂业务领域简单化，帮助我们设计出清晰的领域和应用边界，保证业务模型与代码模型的一致性。

我们都很熟悉从需求细节到设计的部分，却往往遗漏或不去思考最开始也是最重要的东西——业务建模，或者是觉得自有领导去想，或者是认为这不是自己的专业方向，但谁想卖一辈子的糖水呢~

祝心有所梦者，皆能得其所愿。

-End-

原创作者 | 欧霄

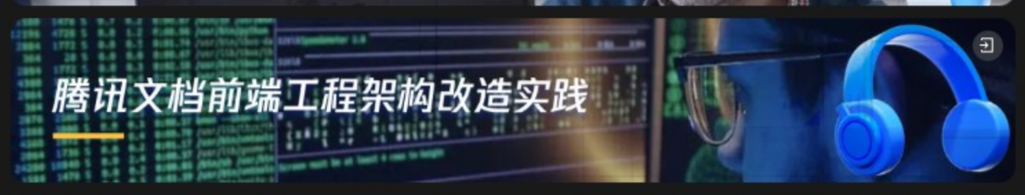
「开发者·开放麦」

你觉得怎么样的架构图是好的呢？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21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
后端开发黑话大全，进来对齐颗粒度！

下一篇
一次网络请求的顿悟之旅