全面解读流程图｜附共享单车摩拜 ofo 案例分析

对于任何产品设计来说，构建流程都是一个绕不开的环节。其奠定了后续的产品框架，是用户体验的基石。本文将从定义和分类出发，结合实际案例，深入浅出地阐述流程图的作用以及画法。

诞生之始

其实，之所以我会写这篇文章，完全是受老曹（人人都是产品经理网——创始人 CEO）的邀（hu）请（you）。他说，对于大多数新人产品经理来说，画流程应该是最基本最必要的技能。但往往大多数新产品人却对流程的概念异常模糊。如果能有一篇全面系统的流程图解析，那一定会受到大家的欢迎，帮助大家快速成长。

我听后深受老曹的感动，一个伟大的 CEO 就应该有这样坚韧的品质。是什么样环境才能造就这样伟大的企业家？我不禁想起了马云第一次创建阿里巴巴时的演讲，那会场上奔跑的影子，是老曹逝去的青春。顿时，我对老曹的佩服有如滔滔江水连绵不绝地涌出，又如黄河泛滥地一发不可收拾。

老曹充满鼓励的眼神深情望着我，然后拿开了架在我脖子上的 40 米长的大砍刀。

我心里默默地回荡着老曹的话：“好好写，写完让你当人人的副总裁，我不骗你。”然后我奋笔疾书，于是有了下面的故事……

定义

流程——顾名思义：水流的路程；事物进行中的次序或顺序的布置和安排。流程是自然而然就存在的，它可以不规范，可以不固定，可以充满问题。

由两个及以上的步骤，完成一个完整的行为的过程，可称之为流程；注意是两个及以上的步骤。

流程图的核心就在于如何排布事物进行的次序，不同的顺序可能造成截然不同的结果。

目的

产品经理画流程图的目的不外乎几点：

流程图为产品设计基石，可以保证产品的使用逻辑合理顺畅
传达需求，用流程图来更好地表达产品逻辑
查漏补缺，检验是否有遗漏的分支流程

分类

流程图以描述对象分类，包括：业务流程图、页面流程图、功能流程图、数据流程图等。

业务流程图（Transaction Flow Diagram, TFD）

先以宋丹丹小品中的一个脑筋急转弯为例：把大象装冰箱，总共分几步？

三步：

第一步，把冰箱门打开；
第二步，把大象装进去；
第三步，把冰箱门关上。

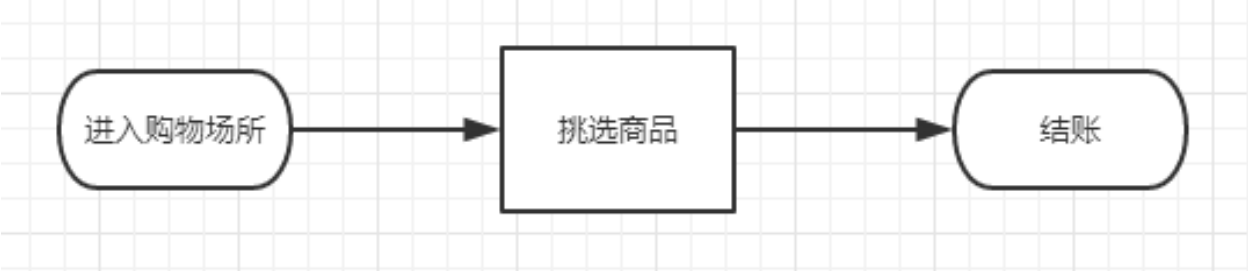
这看似是一个笑话，但其实蕴含着很强的逻辑思维。首先这里忽略了很多现实中的限制条件。比如，以大多数冰箱的容积都不可能将大象塞进去；比如是否能把大象切成块放进去？如果把大象塞进去，它会不会又跑出来？但抛开这些限制条件，那把大象塞冰箱的极简流程就是三步。打开冰箱门，把大象装进去，最后把门关上。

我们做业务流程图，其实很多时候都需要具有把“大象塞进冰箱”的思维方式，抛开很多现有的认知局限，将具象的行为一个个抽象出来。

结合上面的例子，再来细细品味“业务流程图”的定义：

抽象地描述事物进行的次序和顺序，不涉及具体操作与执行细节。在互联网软件行业通常指脱离产品设计的用户行为流程。业务流程图是一种系统分析人员都懂的共同语言，用来描述系统组织结构、业务流程。

不管是否理解上述定义，下面带着抽象思维去思考购物行为的业务流程图应该是什么样的？



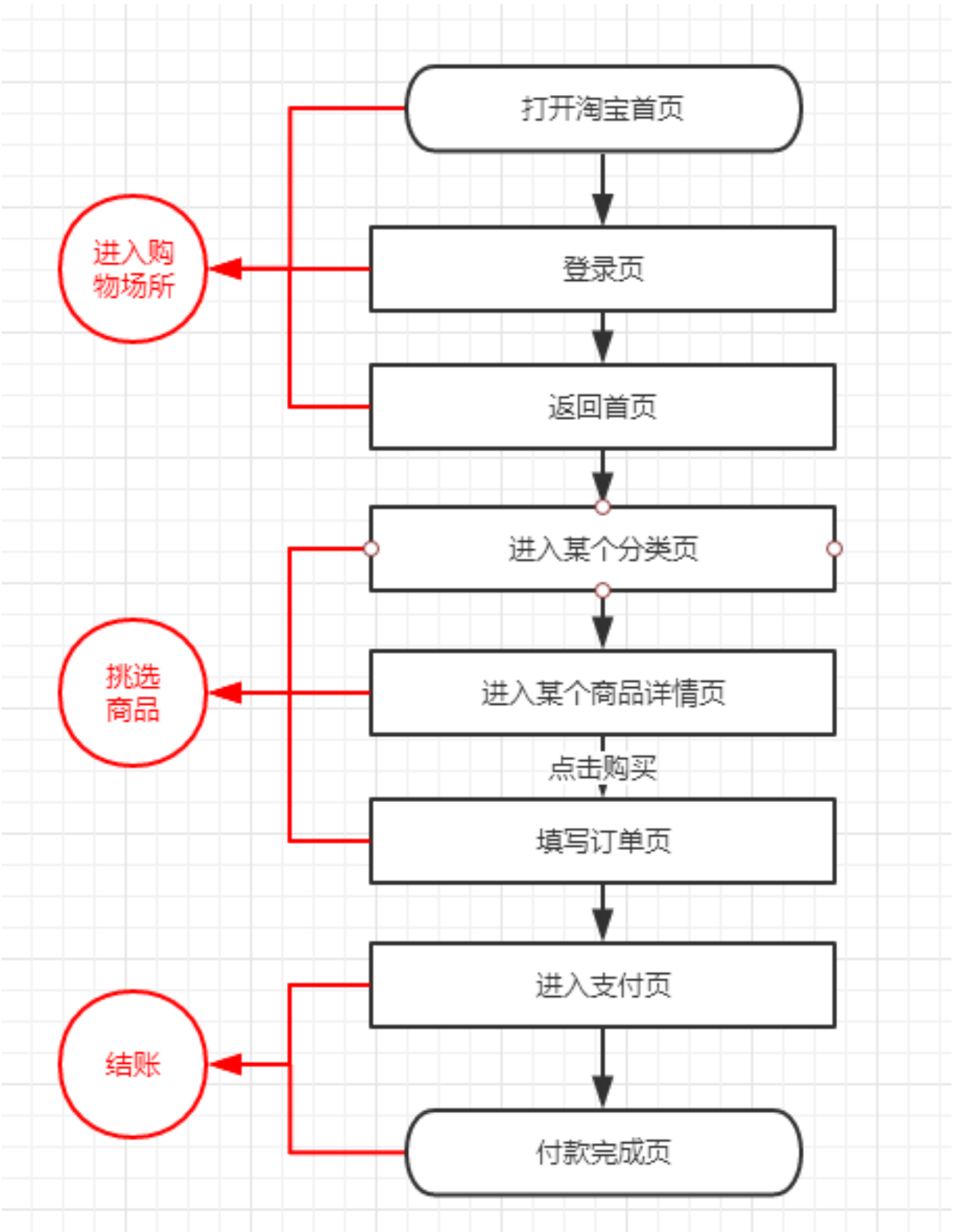
以上的三步组成了一个最简的一个流程，其完全涵盖了任何购物行为的核心。无论是网购还是在实体超市，都是以这三个行为为主体，然后进行扩展的。相对于大家平时看到的复杂的网购流程图，以上的三步流程简直简单的令人发指，而这恰恰是印证了大道至简的原理。我始终坚信无论再复杂的事情都能简化为极其简单的事情，如果你无法将其简化，说明只是你没有理解其核心。

依据上面的最小流程单元，我们下面尝试能不能将其扩展，尝试套用在更细节的流程图上面。

页面流程图（Page Flow Diagram）

定义：指电子产品具体所呈现的页面跳转流程图。其承载了业务流程图所包含的业务流转信息。

下图以淘宝为例，展示出了网购的页面流程。


打开淘宝首页
登录页
返回首页
进入某个分类页
进入某个商品详情页
点击购买
填写订单页
进入支付页
付款完成页
进入购物场所
挑选商品
结账

由上图红框中的三个节点我们可以看出，页面流程图依然是包含在业务流程图的。这恰恰符合定义中的要求，同时也印证了页面流程图的正确性。相较于一开始的极简流程图，现在的流程图已经渐渐变得复杂了一些。我们将抽象的业务，映射在了具象的页面上，用软件的页面承载起了业务需求。而以上就是由业务流程图到页面流程图的转化过程。

功能流程图（Function Flow Diagram）

定义：指单页面内或多页面之间的功能操作流程，其包含在页面流程中。

任何功能都是被包含在页面内的，但一个页面内往往不止一个功能，所以单单页面流程图可能无法完整表达所有流程，而这时就需要用功能流程图来更加具体表达每个页面内所包含的功能。

由上图红框中的四个节点我们可以看出，功能流程图同样也是由页面流程图拓展而来的。功能流程图是在页面流程图的基础上继续深化，变得更加复杂。同时也渐渐变得像大家日常看到的流程图一样。

数据流程图（Data Flow Diagram）

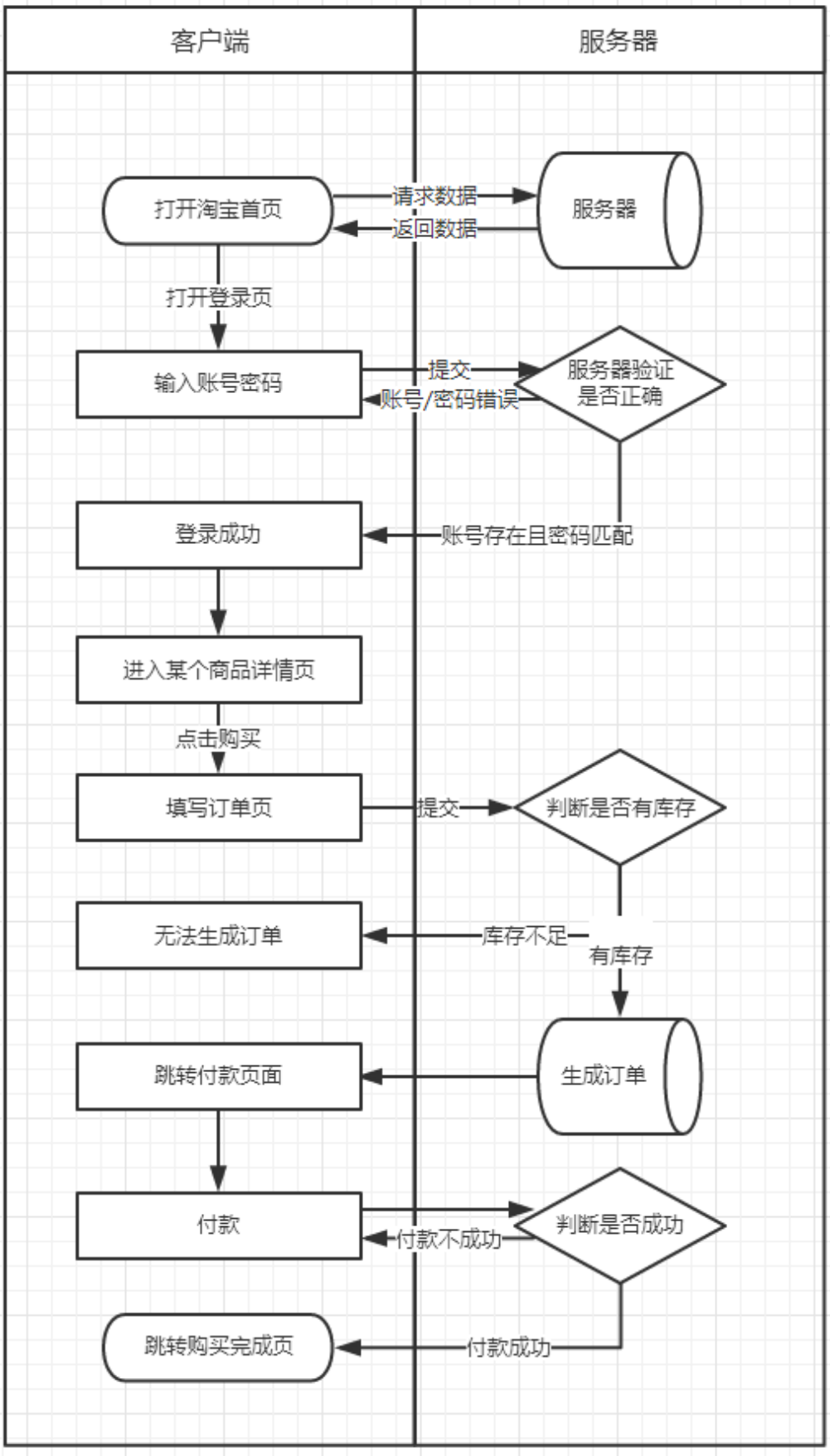
定义：特指软件产品中，描述数据在不同节点被处理的过程所画的图表。主要表达计算机程序对于业务的实现原理。用户在功能流程图中的每一个操作，对应都会反映在数据流程图中。同时，数据流程图也可以叫程序流程图（Program Flow Diagram）。

它是一种能全面地描述信息系统逻辑模型的主要工具。它可以利用少数几种符号综合的反映出信息在系统中的流动、处理和存储的情况。数据流程图具有抽象性和概括性。

可能业务流程图、页面流程图和功能流程图大家都耳熟能详，但数据流程图恐怕了解的就比较少了。其实，每个流程图中都有一个核心伴随着不同操作在整个系统中不断流转。比如业务流程图大多以人为核心，每个节点都是在传递人的不同行为。而页面流程图和功能流程图也类似，都是以人的操作行为为核心，在不同页面和功能间进行流转。但数据流程图不同，它是以数据为核心，展示整个系统中，数据是如何被处理的。

其更偏技术思维，更多的是展现后台程序的实现原理。所以，常常是开发人员绘制此图，而产品经理涉及较少。但随着产品经理地不断成长，向上提高到战略层，而向下则会深入到实现层。理解程序的开发原理和背后的数据流转，无疑会让产品经理对产品设计有更加深刻的理解。

下面仍以购物流程为主题来展示数据流程图。


客户端
服务器
打开淘宝首页
请求数据
返回数据
服务器
打开登录页
输入账号密码
提交
账号/密码错误
服务器验证是否正确
登录成功
账号存在且密码匹配
进入某个商品详情页
点击购买
填写订单页
提交
判断是否有库存
无法生成订单
库存不足
有库存
跳转付款页面
生成订单
付款
付款不成功
判断是否成功
跳转购买完成页
付款成功

相较于之前的图表，数据流程图增加了新的维度——程序。客户端在展现用户操作行为的同时，也表达了程序在用户行为背后的动作。而往往大家说一个产品复杂的时候，可能只注意到了它的前端交互复杂，而忽视了后端逻辑的复杂。对于一个优秀的产品经理来说，不止要关注前端的用户体验，更要能看清事物背后的逻辑。毕竟人人都可以对用户体验指手画脚，而说到程序实现，那可就体现出产品经理的专业性来了。

小结

以上几幅图片分别展示了一个产品的业务流程、页面流程、功能流程和数据流程。从中可以发现，由业务到页面，再到功能，再到数据处理，是顺序拓展的。一个产品的页面或功能，不是凭空出现的，而是依据业务层的各个节点和流程进行设计的。这就是为什么在做产品设计时一定要先理解业务的原因。

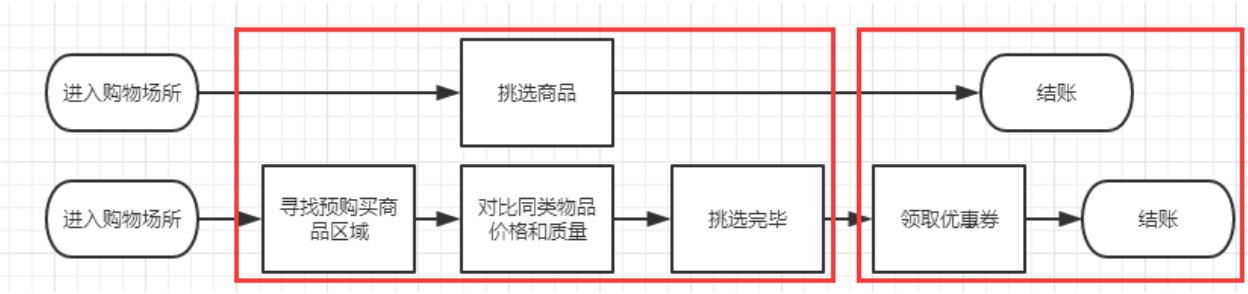
在初步学习画流程图时，尽量将业务、页面、功能和数据区分清楚，并且逐层递进，不要把多种类型的流程图混杂一起。这样反而会将思想搞得混乱。

流程图的颗粒度

所谓流程图的颗粒度，其实就是指流程图的细致程度。

我在画流程图时也常常会犹豫纠结，这个功能点用不用描写得更详细？这条分支用不用标出来？这个和服务器的交互事件用不用在流程图体现？等等这些问题，也都是产品经理在日常画图时会遇到的。

依然拿购物流程为例，最简的业务流程分为三个步骤，那如果细化一些，是否可以画出不同的流程图呢？



显而易见，即便针对同一个流程，也能画出不同的流程图。如上图，将挑选商品拆分为三个步骤，将结账拆分为两个步骤。但两个流程图依然表达的是一套流程。而这就是每个人对于颗粒度的把握有所不同。有很多新人总想一步到位，一次画出完美的流程图。但这其实是一种非常不可取的思维。任何完善的流程图，都需要经过由简单到复杂的过程，而不是一蹴而就。

理论上来说，流程图的细致程度越高，产品设计就越准确顺畅。但实际情况中，过度的详细反而是浪费时间。而对于度的把握能力，则需要经验积累以及团队磨合，这里也是体现产品经理对颗粒度把握能力的地方。我们画流程图的最终目的是让团队成员理解我们的产品设计，而不是需要画一幅非常详细的流程图。理想的情况应该是以最简的形式，画出团队都能理解的图表。

流程图画法

上面讲解了流程图的定义和分类，下面就进入具体的流程画法讲解

流程图基本元素

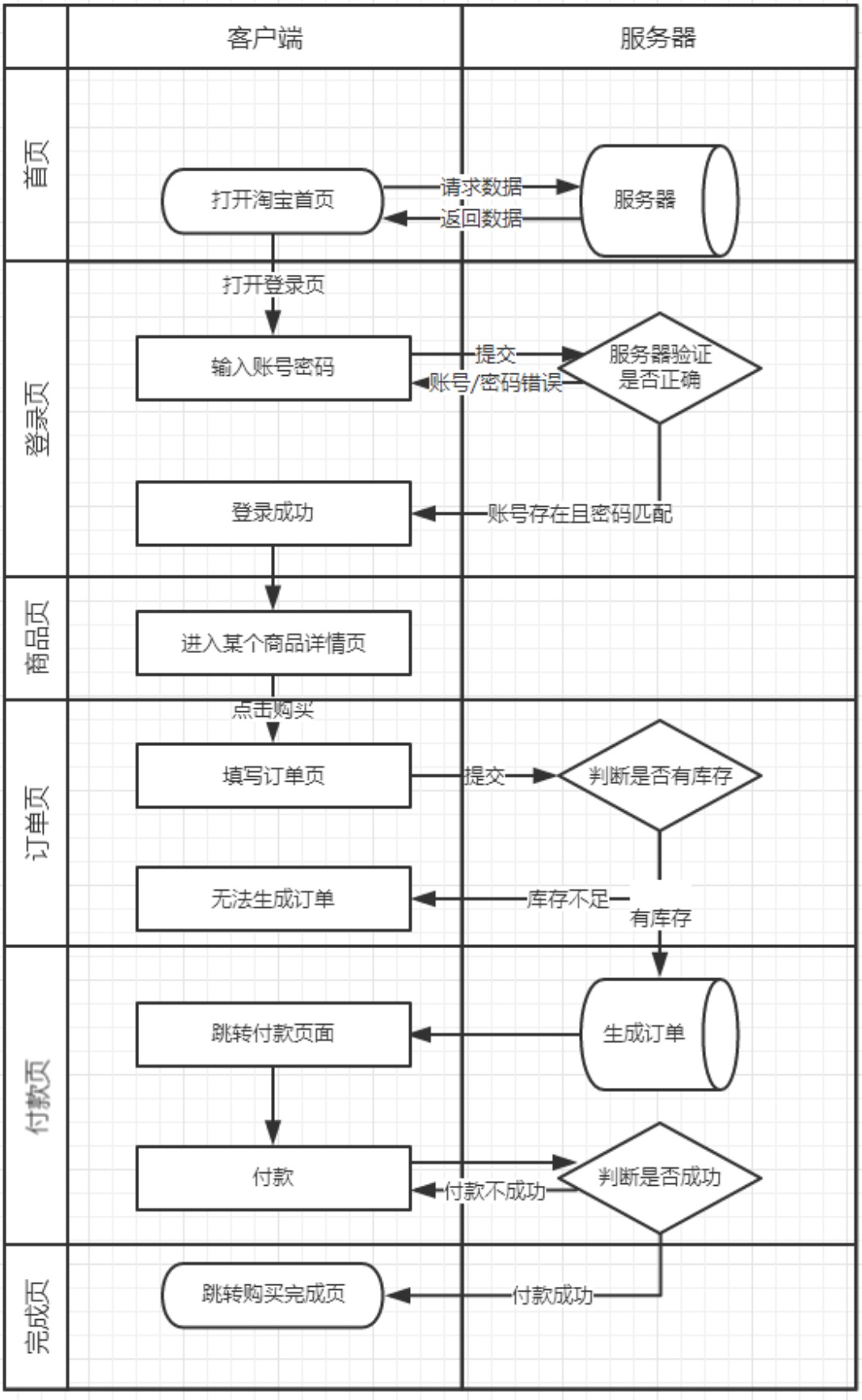
元素样式	元素名称	元素介绍
	开始/结束	流程图的开始或结束都以此元素样式为准
	节点	任何一个操作或者状态，都为一个节点
	判定	遇到不同处理结果的情况下，采用此符号连接分支流程
	子流程	将流程中一部分有逻辑关系的节点集合成一个子流程，方便主流程频繁调用
	连接线	用来将任意节点连接起来，连线上可添加文字

以上为流程图中最常用的几种元素。不常用的元素就不在此展示了，大家可以在 Microsoft Visio 中查看。

泳道图

泳道图是流程图中的一种画法，是将流程图中的一些流程节点按操作角色的不同而划分。比如刚才的数据流程图其实就采用泳道图的画法展示，其中顶部为两个不同角色——用户和服务器。同时在竖向的基础上也可以添加横向泳道，以不同页面来给操作分类。

对于涉及到多角色比较复杂的流程图来说，画泳道流程图会看起来更加清晰明了。

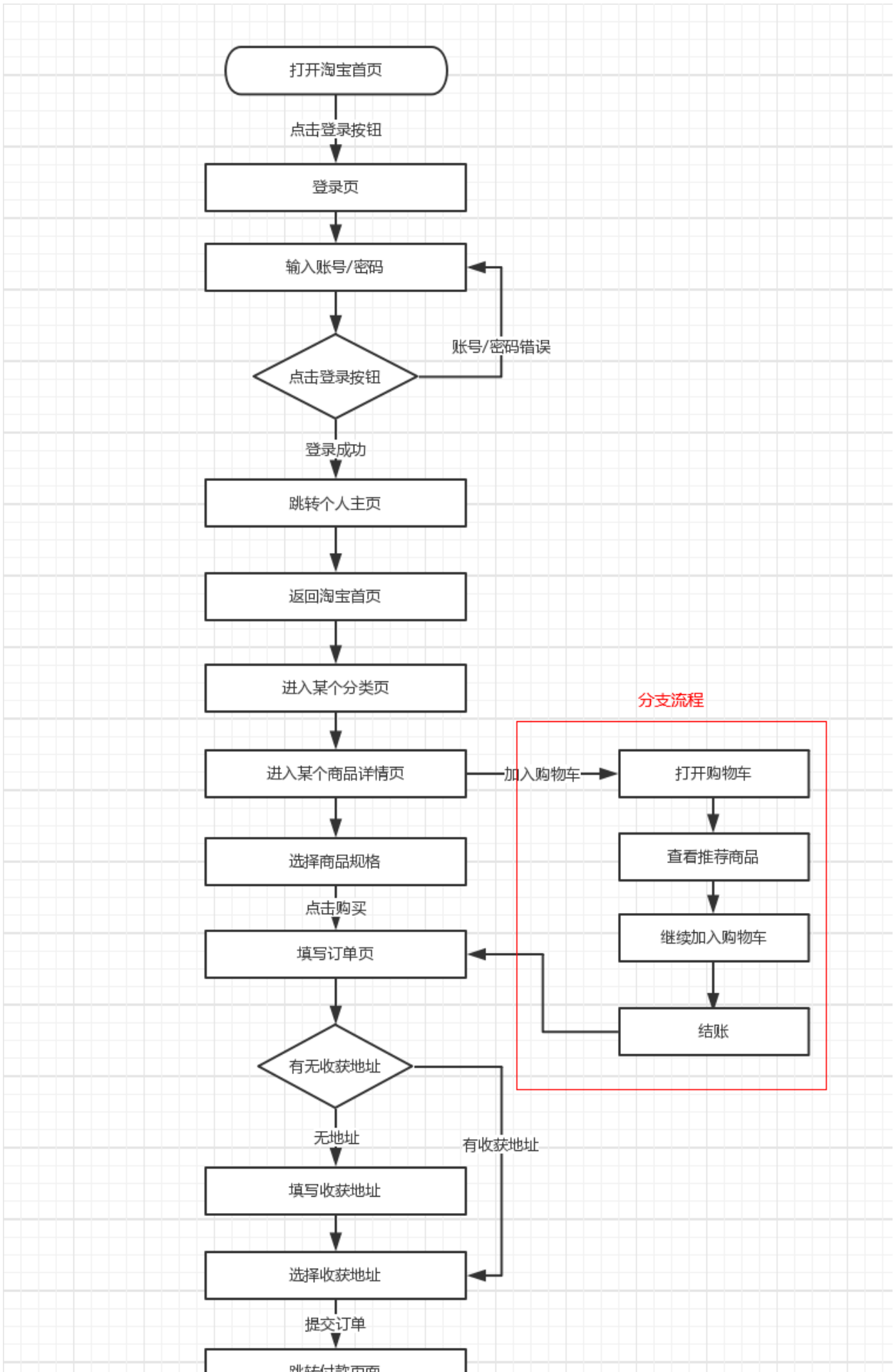

客户端
服务器
首页
打开淘宝首页
请求数据
返回数据
服务器
登录页
打开登录页
输入账号密码
提交
账号/密码错误
服务器验证是否正确
登录成功
账号存在且密码匹配
商品页
进入某个商品详情页
订单页
点击购买
填写订单页
提交
判断是否有库存
无法生成订单
库存不足
有库存
付款页
跳转付款页面
生成订单
付款
付款不成功
判断是否成功
完成页
跳转购买完成页
付款成功

流程图的组成部分

流程图主要由三部分组成：

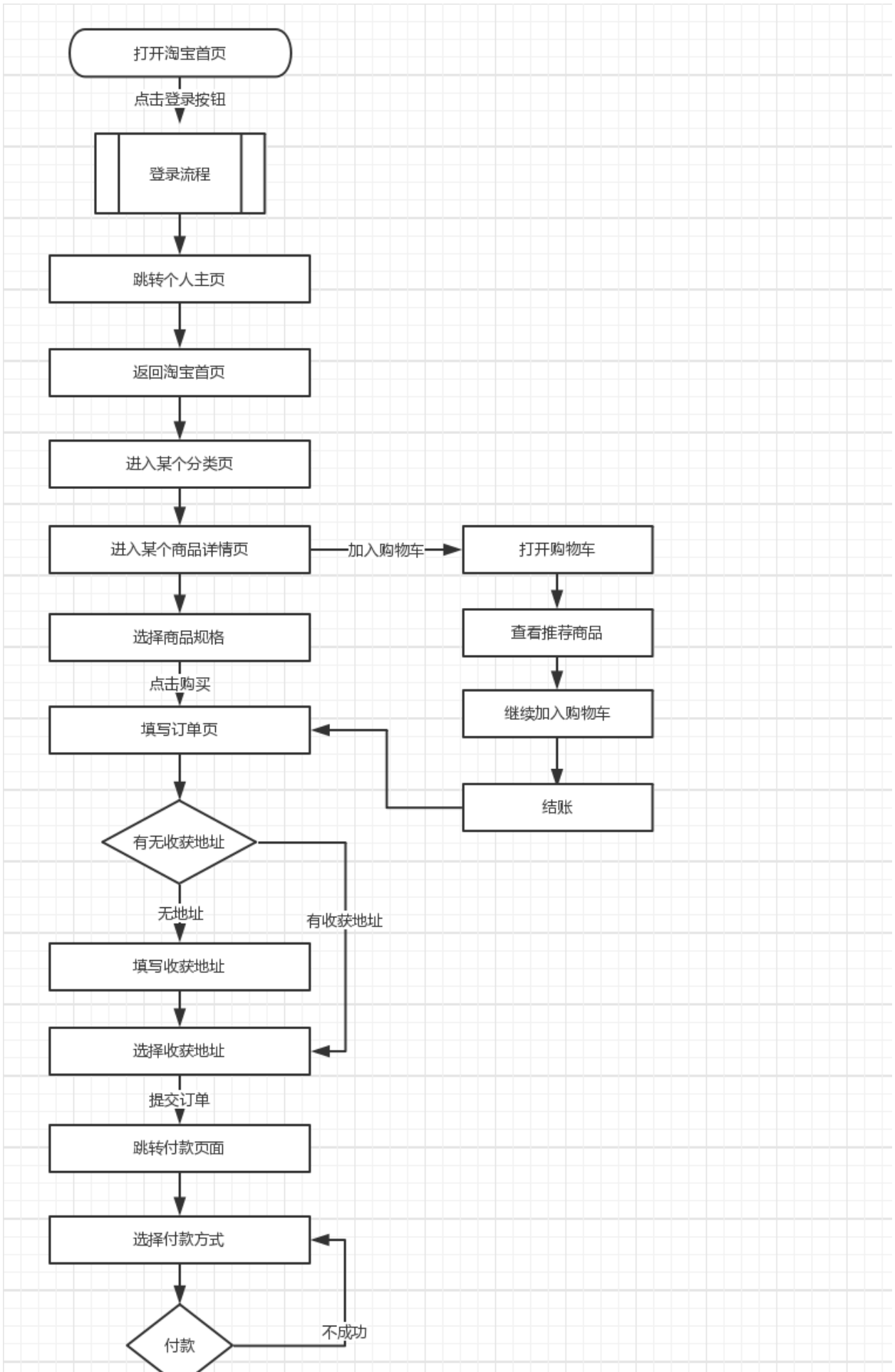
主干流程
分支流程（异常流程属于分支流程）
子流程

下图是将之前功能流程图的例子作为主干流程，然后添加了分支流程。我们在画流程图时应该遵循先主干后分支的顺序来描绘流程图，因为对于大多数用户来说，主干流程是最常用的路径。


打开淘宝首页
点击登录按钮
登录页
输入账号/密码
点击登录按钮
账号/密码错误
登录成功
跳转个人主页
返回淘宝首页
进入某个分类页
分支流程
进入某个商品详情页
加入购物车
打开购物车
查看推荐商品
继续加入购物车
结账
选择商品规格
点击购买
填写订单页
有无收获地址
无地址
有收获地址
填写收获地址
选择收获地址
提交订单

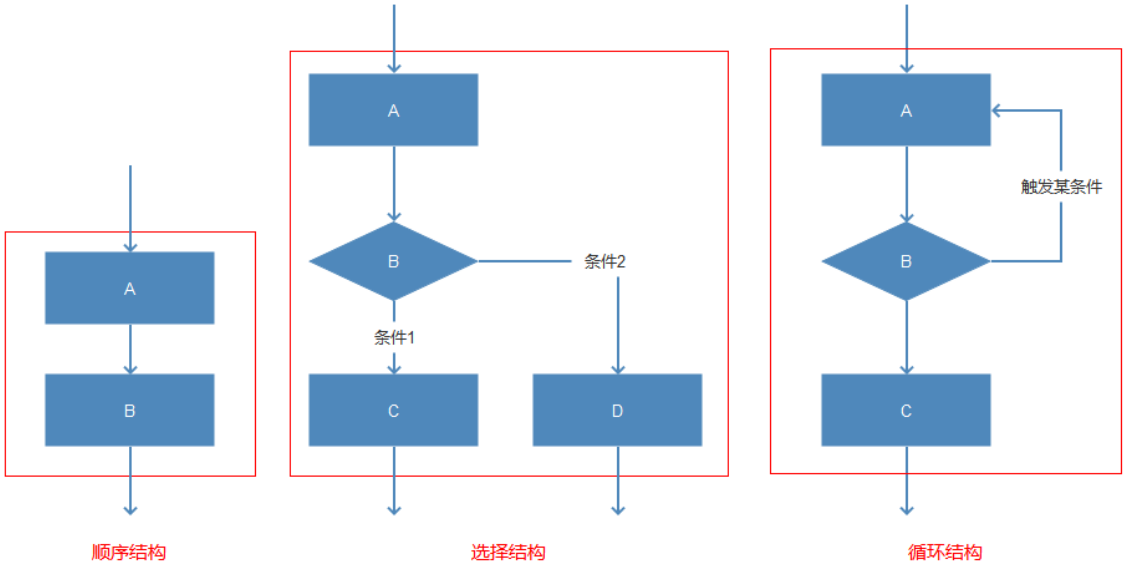
主干流程和分支流程大家都好理解，那到底什么是子流程呢？在画流程图的过程中，有一些流程是会经常遇到的，比如登录流程、注册流程、修改密码流程。对于电商来说，可能有退货流程、购物券使用流程等等。如果每次画与之有关的流程图的时候，都将其再画一遍，那实在繁琐。所以，子流程就是将某几个具有逻辑关系的节点集合而成的，可以复用在各个地方。

下图就是将登录流程变成子流程后的流程图。


打开淘宝首页
点击登录按钮
登录流程
跳转个人主页
返回淘宝首页
进入某个分类页
进入某个商品详情页
加入购物车
打开购物车
查看推荐商品
继续加入购物车
结账
选择商品规格
点击购买
填写订单页
有无收获地址
无地址
有收获地址
填写收获地址
选择收获地址
提交订单
跳转付款页面
选择付款方式
付款
不成功

流程图的结构

流程图中大致包含四种结构：顺序结构、条件结构（又称选择结构）、循环结构。基本上大多数流程图都是由这三种结构组成的。



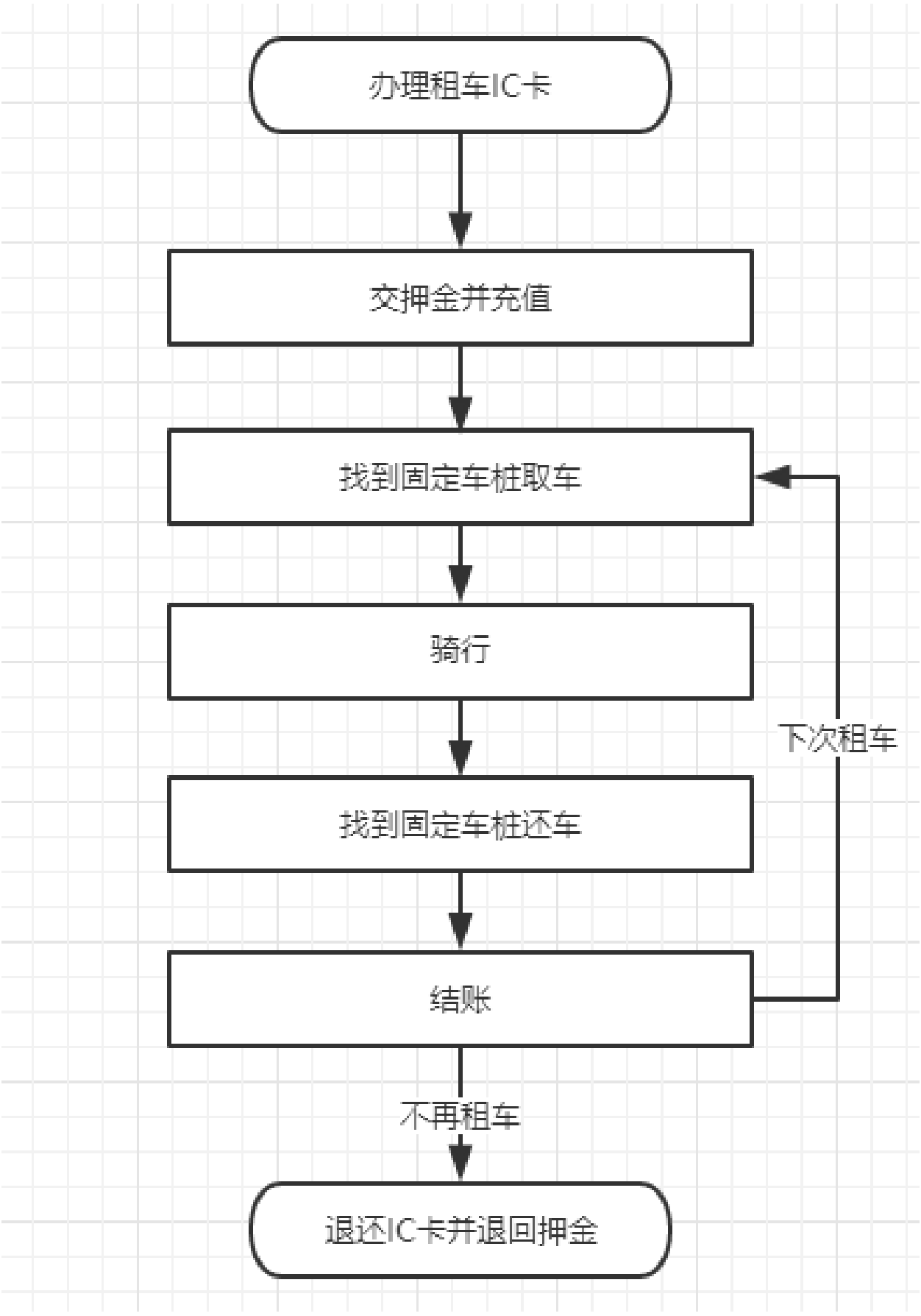
案例

上面说了那么多理论知识和概念，那下面就开始真刀实枪地展示一个案例。本来一开始我想以电商产品作为例子，因为电商产品是需要极强逻辑思维的产品，并且比较常见。但后来发现淘宝、京东等都极为庞大和复杂，分析起来过于笨重。转而想起共享单车是个非常不错的教材案例。其产品极简，但背后却暗藏有趣的逻辑架构。尤其是市面上摩拜与 ofo 不同的产品解决方案，分析起来更加有对比性。

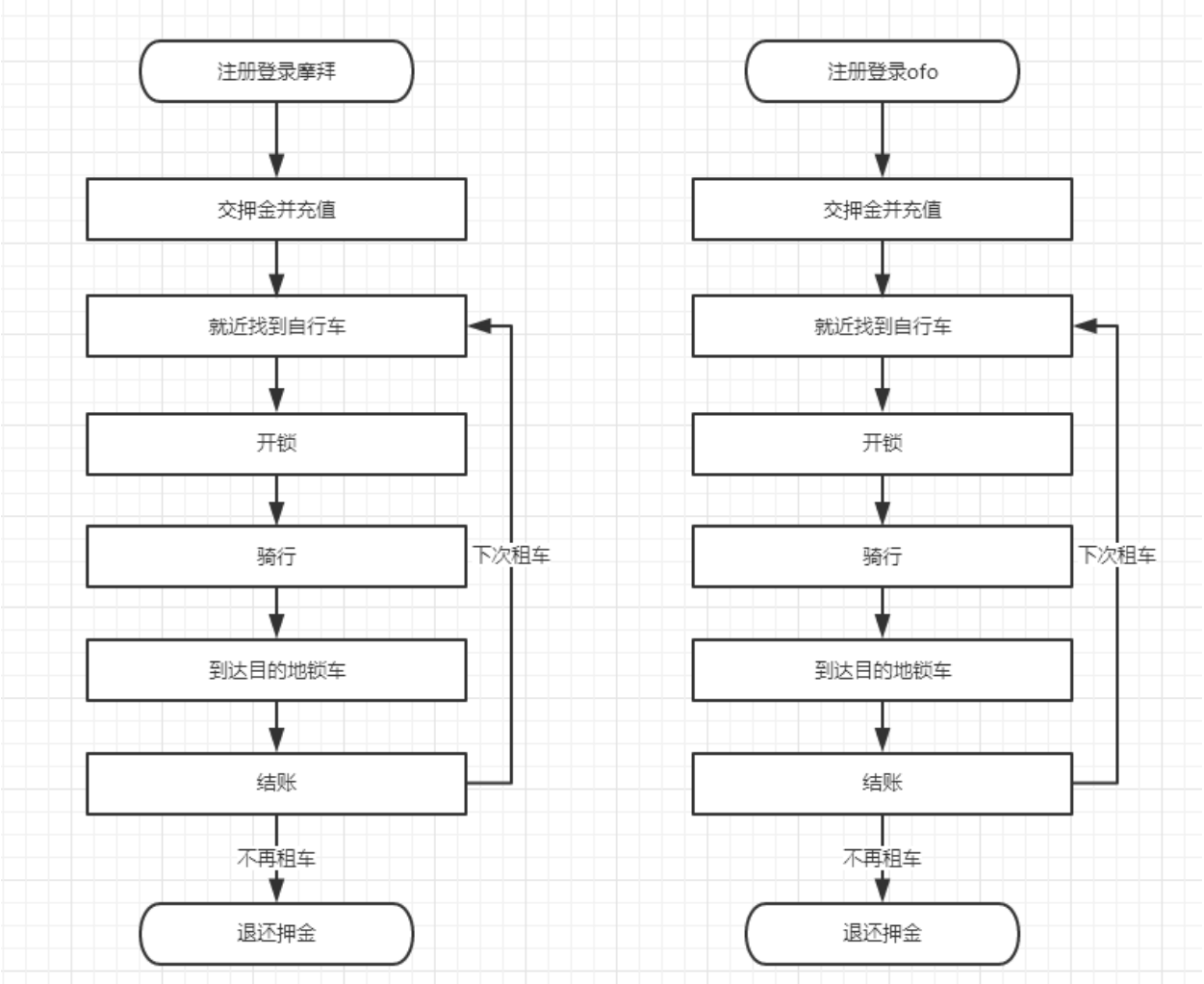
共享单车的前身

如果要追溯最早的共享单车，恐怕就是政府推出的有桩自行车。其推出目的无非就是缓解交通压力，以及减少环境污染。而当时受限于成本、技术以及大众人群的普遍素质，有桩自行车的解决方案是极其不方便的。想要租一辆有桩自行车，首先要凭身份证在相关单位办理 IC 卡，并缴纳押金和预存费用，然后租车和还车只能在定点位置进行。先不谈办理卡片有多麻烦，租车还车有多不方便，超时扣费有多惊人，如果只单纯将其用业务流程图展示出来，应该是什么样的呢？

下面依然以最简单的业务形态来展示使用有桩单车业务流程图：


办理租车IC卡
交押金并充值
找到固定车桩取车
骑行
找到固定车桩还车
结账
下次租车
不再租车
退还IC卡并退回押金

单看有桩单车的流程图其实没有任何意义，真正的意义在于有桩单车和目前摩拜与 ofo 的横向对比，下面看一下两家共享单车的业务流程图：



很明显可以看出，无论是有桩单车、摩拜单车还是 ofo 单车，在业务流程图上竟然没有太大区别。那为什么多年之前政府主导的有桩单车平平无奇，而 2016 年末出现的共享单车红极一时？那摩拜和 ofo 两款截然不同的单车，区别点到底在哪里呢？我们需要更加深入地分析每个业务节点，剖析其功能。

因为单车的使用流程不仅是在 APP 上，还有一部分操作在实体自行车上，这时就不能单使用页面流程图，而是要直接使用功能流程图。并且这里的功能流程图不局限于页面内的功能，而是要表达用户对单车和 APP 的每一步操作。

首先看 ofo 单车，在 APP 中支付押金后，接着便需要寻找自行车。而这时我们发现，虽然 ofo 有多种单车样式，多种车锁机制。但本案例着重讲 ofo 第一代机械锁，与第二代伪智能锁。

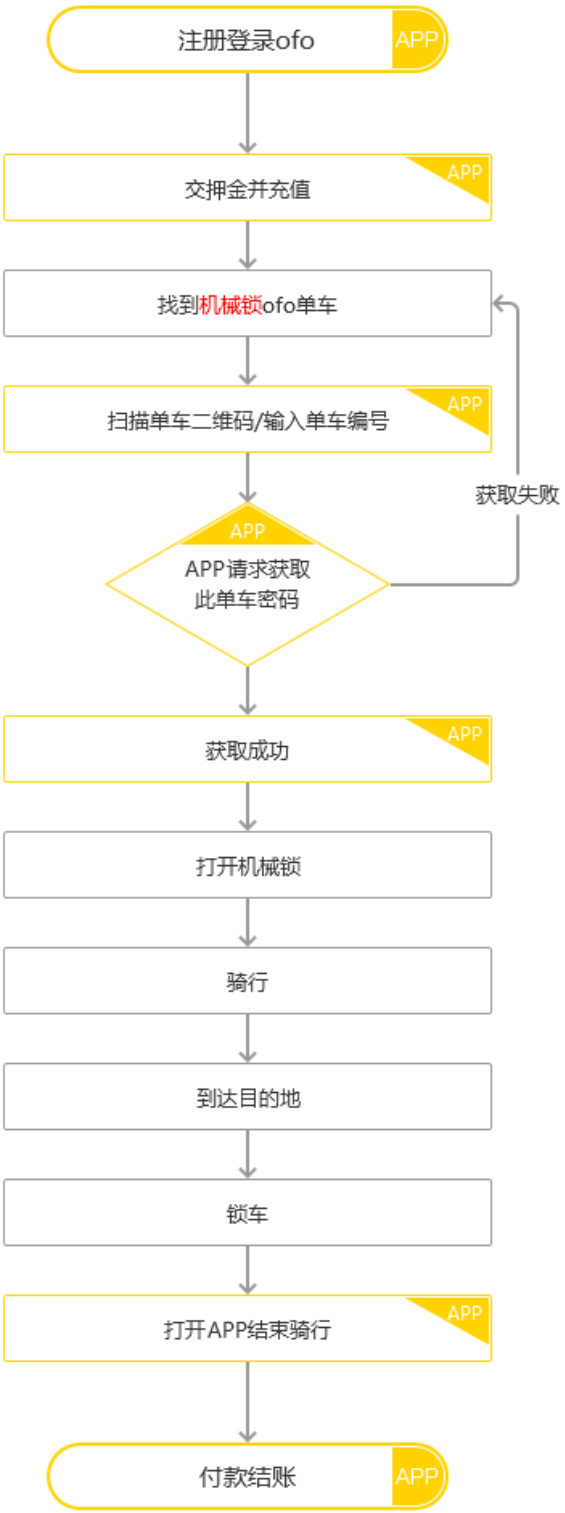
这两种锁其实代表了两种不同的产品解决方案，我们先讨论第一种机械锁。（所谓机械锁，其实类似于生活中经常见到的密码箱。每个密码箱有预设的固定密码，通过拨弄表盘输入正确密码，即可开锁。并且机械锁的密码是固定的，不会改变）。

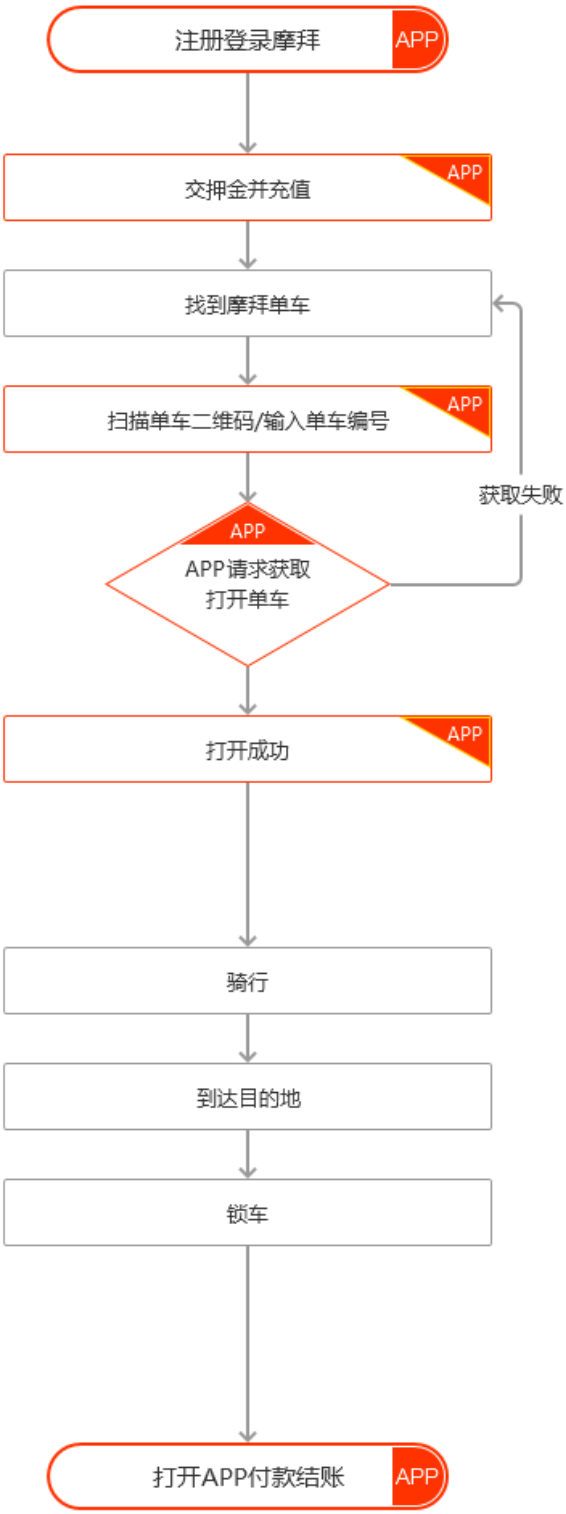
我们从路边找到机械锁单车，然后打开 APP，输入车牌号或扫描二维码，从 APP 中得到本车的机械锁密码，然后输入密码，打开单车车锁。此时 APP 中会进行倒计时，倒计时结束则开始正式计费。最后，骑行到目的地后，需要将车锁关闭，并且必须在 APP 中点击结束骑行的按钮，才能结算此次行程的订单。

看完了 ofo 的流程，下面来对比看一下摩拜的流程。

摩拜的产品解决方案为，扫描单车的二维码以后，摩拜单车的车锁会自动打开，不需要像机械锁一样手动操作。并且在锁车后，摩拜单车自动会结束行程，无须在 APP 中点击结束。在下一次 APP 打开时，才会进行账单结算。

下图分别为 ofo 机械锁单车使用流程图和摩拜单车使用流程图（APP 标识代表用户在 APP 上的操作）





我们可以清楚地看到摩拜的流程比 ofo 的少了两个节点，而这就是摩拜对比 ofo 第一代机械锁的优势。当然，ofo 第一代也有其他方面是优于摩拜的，比如骑车的舒适程度。但本文主要聚焦于产品流程，所以并不在单车体验上花费太多笔墨。

纵观 ofo 机械锁和摩拜智能锁的解决方案来看，ofo 明显是逊色很多的。机械锁带来的问题，不止是使用流程的复杂，还有很多是产品使用上的漏洞。比如，用户锁车后，必须手动将密码拨乱，不然下个人将可以免费骑行。比如，用户在骑行结束后，忘记在 APP 点击结束，会造成更额外扣费。等等还有很多问题，就不一一列举了。

说句题外话，这些问题 ofo 也都明白。机械锁的解决方案倘若只在封闭的校园内运行，那还差强人意。但一经投放到校外市场，那么这种解决方案无疑会给公司带来巨大的损失。那为什么 ofo 明知问题，还要大量投放呢？原因很简单，以摩拜拓展的速度，如果他不在当时迅速走出校园，那么也许永远也没机会走出校园了。

言归正传。之前的讨论，一直避开了一个非常重要的节点——“找车”。抛开路边随机看到单车不谈，就拿地图找车来说，ofo 第一代机械锁肯定是没有 GPS 定位的，为什么也能在地图上显示呢？

下面我们尝试画一下 ofo 对于解锁的程序流程图是什么样的。

我们从“APP 扫描二维码/输入单车编号”此节点开始推导。我要开车牌号为 XXX 的单车，那么就需要得到密码，而所有车的密码，都应该放在 ofo 的单车数据库中。我们不论是扫描二维码，还是输入单车号，本质都应该是将单车编号传输给服务器，告诉它我要哪辆车的密码。服务器查询到此单车的密码以后，就传输回 APP，我们就看到了此单车的密码。

因为节省车锁电源的原因，服务器此时并没有和单车联系，而是靠人工输入密码打开车锁。所以 ofo 在用户得到密码后，就会开始倒计时。倒计时内可以取消开锁状态；倒计时结束，则代表用户默认开始骑行，计费也从此时开始。

此时如果是 iPhone 用户的话，将 ofoAPP 最小化时，就会发现手机顶部电池电量条变成了蓝色。其实，这就是 ofo 获取单车行程的要点所在。既然机械锁无法向服务器传输数据的话，那不如让用户手机代替。以获取手机的定位来获取单车的骑行路线。并且在停车后，点击结束骑行时，上报位置，由此服务器来标记此单车停放的位置。而此时上报的位置其实并没有单车。这就是 ofo 地图上有很多假标记产生的原因。


●●●○○ 中国移动 4G 12:26 100%
“ofo共享单车”正在使用您的位置信息
星期日
6
日历
时钟
照片
相机
附加程序
电商
App Store
设置
信息
百度阅读
QQ邮箱
锤子便签
文件夹
支付宝
微博
知乎
知乎日报
起点读书
美拍
弹琴吧
5周年
部落冲突
Keep
1
有道云笔记
UC

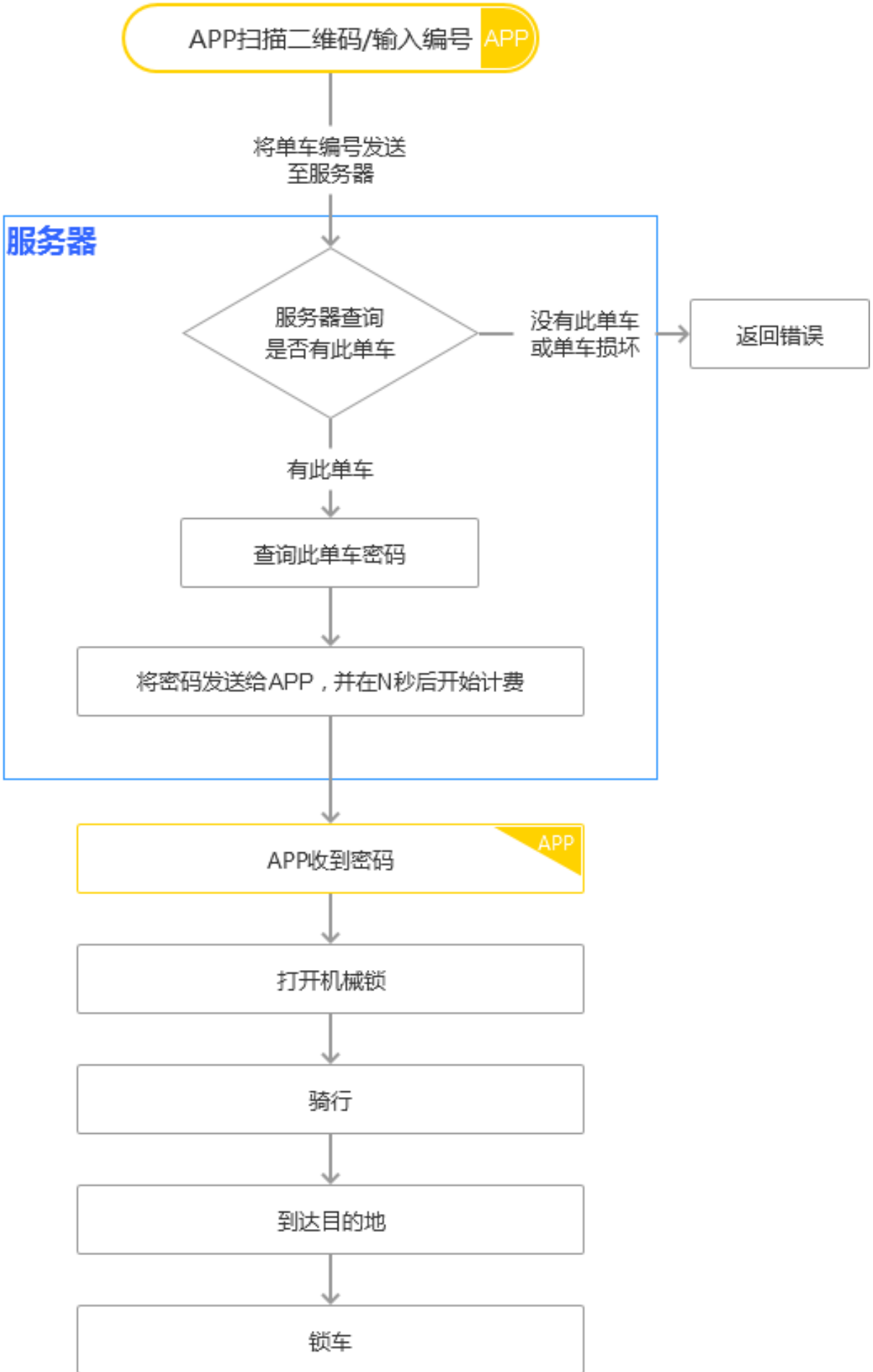
ofo 采用的这种标记方法其实非常的粗糙，毕竟如果用户强制结束应用，也就获取不到骑行路线了。而 ofo 针对获取不到骑行路线的情况，也做了处理，那就是用标记起点到终点，然后根据地图提供的路线来显示路程。

呆萌的李狗蛋儿

¥0.00 ¥100 09分45秒 12142462

上图我亲测的案例。红色箭头是我的实际骑行的路线，绿色线是 ofo 自带地图上通过起点和终点计算的路线。

下面我们继续分析 ofo 机械锁的程序流程图


APP扫描二维码/输入编号
APP
将单车编号发送
至服务器
服务器
服务器查询
是否有此单车
没有此单车
或单车损坏
返回错误
有此单车
查询此单车密码
将密码发送给APP，并在N秒后开始计费
APP收到密码
APP
打开机械锁
骑行
到达目的地
锁车

注意上图服务内的部分，看起来步骤非常少，也非常简单，而真实的服务器肯定有更多复杂的逻辑判断。但对于产品经理画的流程图来说，不可能完完全全描绘编程中的技术细节，而且也不需要产品经理去帮技术想代码的实现逻辑。我们要做得是，理解程序宏观的实现逻辑。

比如，在扫描二维码后，为什么 APP 会显示这辆车的密码，而不是其他车辆的密码呢？很简单，服务器内肯定储存了所有单车的密码，而扫描二维码的过程就是将此单车的 ID 传送给服务器，服务器在数据库中找到密码后，返回给用户手中。

服务器在此处理过程中，肯定还会有其他的判断，比如此用户账号是否正常，有没有被封号？此单车是否已被标记为故障车？等等。但大家发现，上面的流程图内并没有画出这些逻辑判断，是我忘记了吗？

其实并不是。这里又不得不提到本文的核心概念——颗粒度。

此图想表达的是宏观的程序实现逻辑，是为了让读者更聚焦于问题核心，我们只需要着重表达主干流程就好。如果添加更多的分支流程、异常流程，那反而会影响读者的注意力。所以，还是老生常谈的那句话：画流程图一定要先主干，后分支，千万别在一开始就盲目追求细节。

言归正传，ofo 的第一代锁的解决方案虽然漏洞百出，但依然用其巧妙的方式，实现了地图上单车位置的显示。ofo 推出的第二代锁，改进了以往机械锁的很多问题。其中最大的效果就是车锁的密码不再是固定的，并且锁车之后，不需要再点结束行程。那既然 ofo 的锁已经优化了，那为什么前文还称他为伪智能锁，他和真正的智能锁差在哪里呢？为什么 ofo 的车锁依然需要手动输入密码，而不是像摩拜一样，车锁直接弹开？为什么常常在地图上看到有车，而实际地点没有车呢？

下面引入一个 80、90 后童年的回忆：将军令。

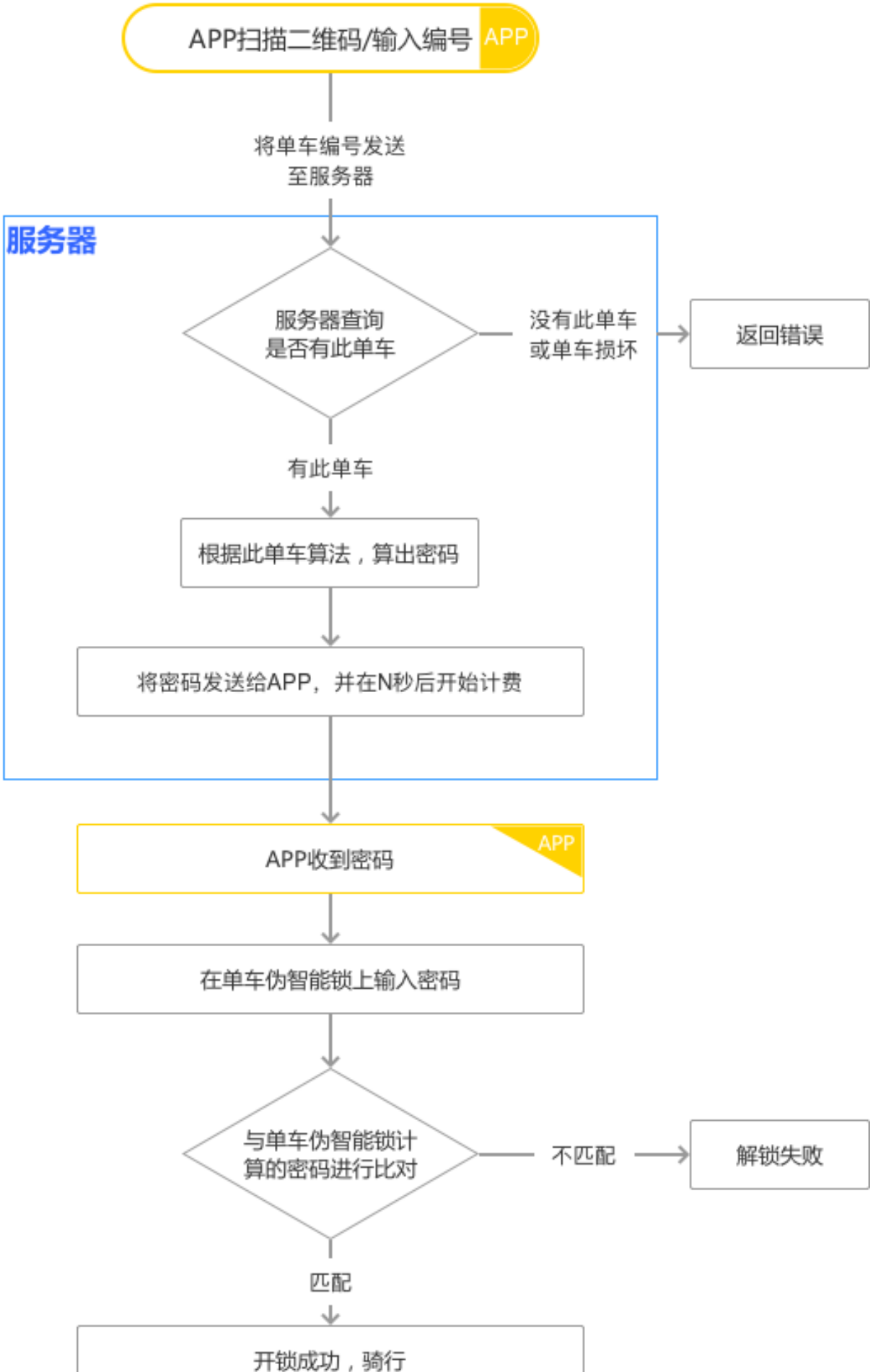
“将军令”（又名网易帐号保护器） 是广州网易互动娱乐有限公司自主研发的、具有完全知识产权的高科技身份认证产品。它是专为保护网易通行证账号（游戏账号）、直销商帐号的密码而出的产品，其特有的 60 秒密码动态自动更新技术将盗号风险降到最低。“将军令”的产生伴随着当年梦幻西游的风靡，其创新技术确实解决了大多数盗号问题。那将军令的实现机制到底是如何呢？简单地说明一下：首先，打开“将军令”，它会生成一串数字，你在登陆游戏时，输入这些数字，系统就会允许账号登陆。同时，“将军令”的数字是每隔 60 秒动态变化的，每次登陆时，“将军令”的验证码都会不一样。其实现原理，无非是“将军令”和服务器保持同一种算法，在同一时间，他们的计算结果是一致的。

回来看 ofo 的伪智能锁，其实也是一样的实现原理。每个车锁在内部保存一个算法，这个算法在服务器内也保存着一份，车锁每隔一段时间就会根据算法变换一个密码。而当你打开 APP，查看此单车密码时，服务器使用已保存的算法算一遍当前时间下的密码，那此密码一定是和车锁当前算的是一致的。这就是 ofo 伪智能锁的开锁原理。

开锁说完了，下面聊聊关锁。如果你骑过小黄车就会发现，小黄车的智能锁关闭以后，并不用手动点 APP 上的结束行程了。那要做到这点，一定是车锁与服务器进行了通信，告诉服务器用户已经结束了行程，可以结算订单了。那既然车锁可以和服务器通信，为什么 ofo 还要采用上面的将军令方式来解锁呢，为什么不直接用服务器通信告诉单车自动开锁呢？

其实，就 ofo 第一代伪智能锁“海王星”来说，并没有做到实时和服务器通信。在关锁的时候，只是车锁单方面向服务器发送消息。而同时服务器收取到消息后，在地图上显示其单车的定位。我想 ofo 这么做，一定是因为想减少车锁的耗电量。要知道实时通信是非常耗电的。

下图是 ofo 的程序流程图


APP扫描二维码/输入编号
APP
将单车编号发送
至服务器
服务器
服务器查询
是否有此单车
没有此单车
或单车损坏
返回错误
有此单车
根据此单车算法，算出密码
将密码发送给APP，并在N秒后开始计费
APP收到密码
APP
在单车伪智能锁上输入密码
与单车伪智能锁计
算的密码进行比对
不匹配
解锁失败
匹配
开锁成功，骑行

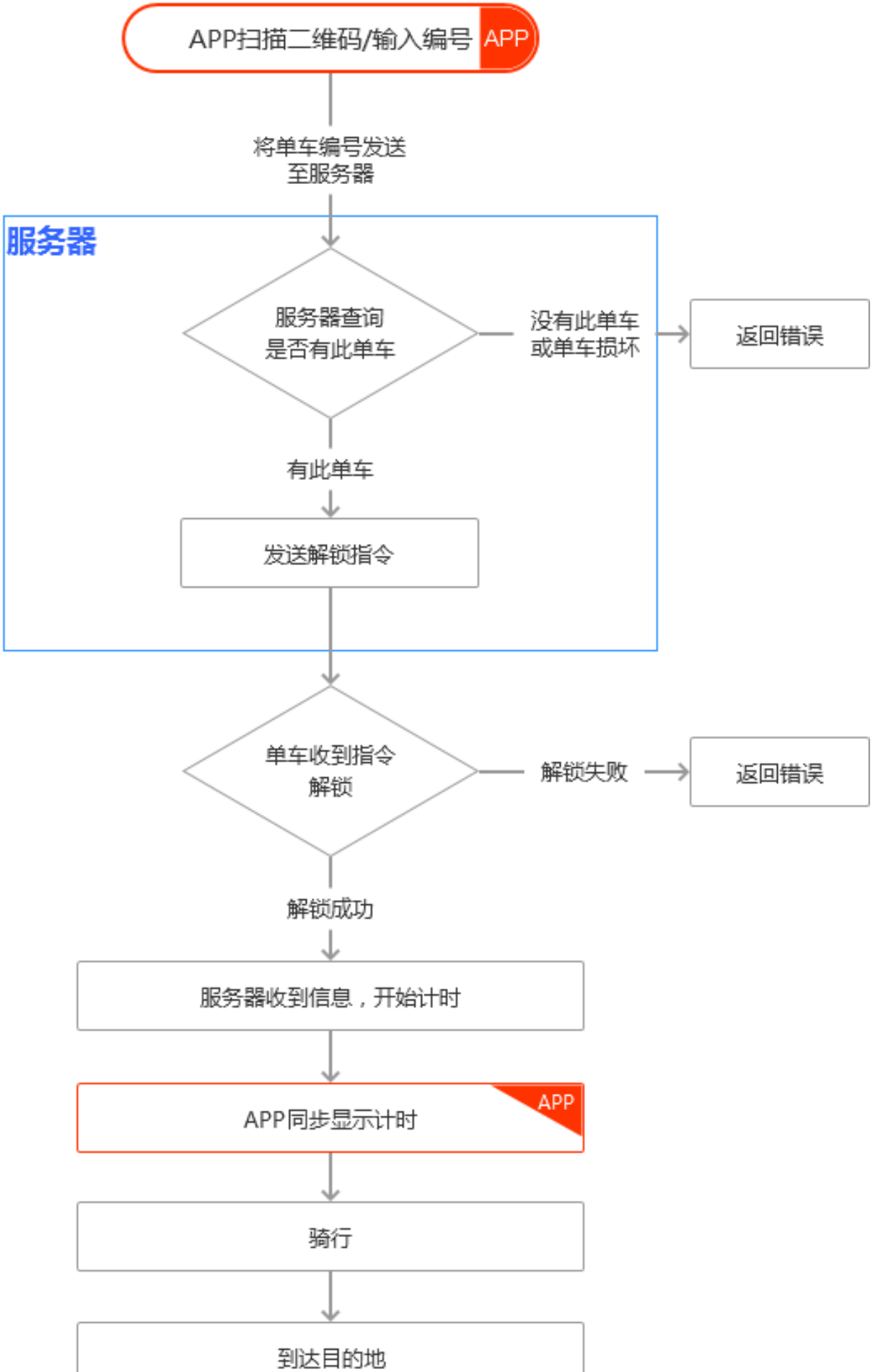
摩拜单车的智能锁

上面分析了 ofo 的机械锁和伪智能锁以后，我们再来看看摩拜单车的智能锁，到底智能在哪里。

首先通过实际体验我们知道，摩拜单车是不需要输入密码的。抛开蓝牙本地验证密码的方式，那摩拜车锁需要和服务器实时通信，才能实现 APP 扫描之后自动打开。

可能有些读者不明白为什么一定要实时通信呢？难道在开锁时，服务器给车锁发送打开的指令不行嘛？举个例子，手机开飞行模式的时候，是无法接听电话和数据上网的。手机想接听电话和上网，就一定要每时每刻和通信基站连在一起，这样才能保证通信基站想要找你的时候可以找到。如果你和通信基站断开，基站是无法知道你位置的。但如果你想要连接基站，只需要关闭飞行模式，打开信号，就可以和基站重新连接起来。这就是摩拜单车需要实时连接服务器的原因。只有这样才能实现单车在地图上的定位以及扫码开锁。其实摩拜为了实现此功能也是大费周章。因为车锁比较耗电，传统电池是无法支撑的，所以摩拜的车采用了机械发电的原理。只要有人骑车，就会将机械能转化为电能，给车锁充电。这也解释了为什么摩拜的车比较难骑，阻力大，车身重。

下面依据上述原理，画出摩拜单车的程序流程图


APP扫描二维码/输入编号
APP
将单车编号发送
至服务器
服务器
服务器查询
是否有此单车
没有此单车
或单车损坏
返回错误
有此单车
发送解锁指令
单车收到指令
解锁
解锁失败
返回错误
解锁成功
服务器收到信息，开始计时
APP同步显示计时
APP
骑行
到达目的地

由上图对比 ofo 的流程，可以看出摩拜采用的解决方案是将自行车与服务器连接。让每一个自行车都成为一个终端，实时同步在整个地图上面。这样既获得了良好的用车体验，也收集到了用户数据。就解决方案来看摩拜是比 ofo 完善很多的。

单车车锁的蓝牙解决方案

大家在用摩拜或 ofo 时，可能经常会看到提示：用蓝牙解锁更加快捷方便。那其实现原理是什么样的呢？我们不妨推测一下。首先，用户在打开蓝牙后要让单车解锁，那就一定要和单车连接起来，否则不可能实现解锁。那单车的蓝牙就必须是实时打开的状态，以供任何时候用户进行解锁。那这时又有一个问题，如果周围有很多单车，那我的蓝牙到底要和哪辆单车匹配呢？这时就体现出扫码的作用了。我一定是扫码的时候，告诉服务器：我要解锁 XX 编号的单车。那服务器会返回给你这个单车的蓝牙”口令”，你通过“口令”与附近的蓝牙匹配，能匹配成功的一定是你想开的那辆自行车。因为你手机的蓝牙和单车的蓝牙距离非常近，蓝牙匹配是非常快速的。所以，通常摩拜或 ofo 都会推荐大家使用蓝牙解锁，这样的成功率更高，速度更快。就蓝牙解决方案来说，ofo 和摩拜其实没有太大区别的。

至于蓝牙解决方案的流程图，就交给大家当作本文的作业。如果你想检验一下本文对你到底有没有帮助，那么你可以尝试去画一画蓝牙解决方案的流程图。相信我，非常简单的。

以上就是整个 ofo 与摩拜解决方案的对比，其中我也画出了不同阶段的流程图。基本可以代表我分析案例的一些思路。最主要的还是让大家能够理解并应用流程图到日常产品设计与分析中。我们在构建流程图时，如果能按照本文的方法，由业务到程序，由简单到复杂，那相信一定会让自己的思路更加清晰顺畅。

总结

本文从定义、分类以及画法，分层讲解了各种流程图的特点。尝试以教科书的方式来阐述其原理和机制。因为目前并没有统一的流程图规范，所以文中难免有错误和理解偏差，也希望大家能指证与交流。

虽然本文的目的是介绍流程图，但其整个思维过程才是真正我想表达的核心。任何复杂事物都可以拆解为最小单元，然后由最小单元逐渐还原复杂事物。引申下去，这种思维方式其实是一种剖析事物的思维模型，熟练掌握以后可以套用于多种分析场景，希望以后有机会单独写一篇文章来详细介绍思维模型。

附件下载

如果你想下载文中涉及的流程图案例的源文件，请关注公众号起点学院（ID：qidianxueyuan666）回复关键词“流程图”即可免费下载。

参考引用：

流程_百度百科
产品经理业务流程图的绘制流程分享
如何绘制业务流程图
谈谈页面流程图（附案例）
产品设计流程系列：业务流程和流程图介绍
众云行第二弹：从 BRD 到页面流程图
数据流程图_百度百科
解读 NB-Iot 智能锁：为何 ofo 和摩拜都要做 NB-Iot 智能锁？

#专栏作家#

作者：臻龙，人人都是产品专栏作家，全面发展型产品经理，喜欢认知心理学与神经科学，特别钟爱讨论抽象算法问题，欢迎撕逼交流。联系 QQ：253884135