业务分析与项目管理

目录

一、业务场景介绍

1.1 为什么要提出业务场景

在企业当中，技术是服务于公司业务，是企业为客户的业务场景通过使用技术进行赋能的一种手段。在一款软件研发的初期阶段，企业会由专门的业务人员对用户的需求进行调研和分析，再根据提炼出的详细需求进行产品功能的设计以及技术架构的选型，之后才进行代码的开发工作，而调研和设计环节往往能占到整个软件研发比重的 50%甚至更多，由此可见，需求来源于场景，软件的价值需要通过实现满足用户的需求来完成生产力到商业价值的转化。此外，一旦在需求分析和产品设计阶段出现偏差，例如无法识别伪需求，功能不能满足用户需求等，种种原因都会导致开发阶段的工作事倍功半，这也是部分不重视业务场景企业的程序员加班屡见不鲜的重要原因之一，极大浪费了企业宝贵的资源。

综上所述，一款产品的成功，技术固然是核心竞争力，但在技术壁垒相差不大，产品功能同质化竞争的今天，产品力已然成为了另一项企业制胜的关键法宝，也就是说，光有技术还是不够的，技术实际运用的场景才是关键，我们学习这些技术就是为了知道它们在实际场景中的用途，在怎么样的条件下才能以最小的成本从而实现最优的效果，最终通过提升用户满意度来为企业和个人赚取更多的利润，而不是盲目的堆积各种技术或钻牛角尖导致最后错失宝贵的市场机会！

1.2 什么是业务场景

业务场景是指在特定时间和空间内发生的行为，可以通过“谁”，“在什么环境下”，“为了解决什么问题”，“如何互动”，“产生了怎样的价值”来进行清晰且详细的刻画。通过这样的描述，商家和消费者，企业和客户，平台商和供应商，在时间和空间两个维度上进行了有效连接，简单来说业务场景就是基于这两个维

度并包含了人、事、物等相关要素的集合。

二、需求介绍

2.1 需求的定义

2.1.1 什么是需求

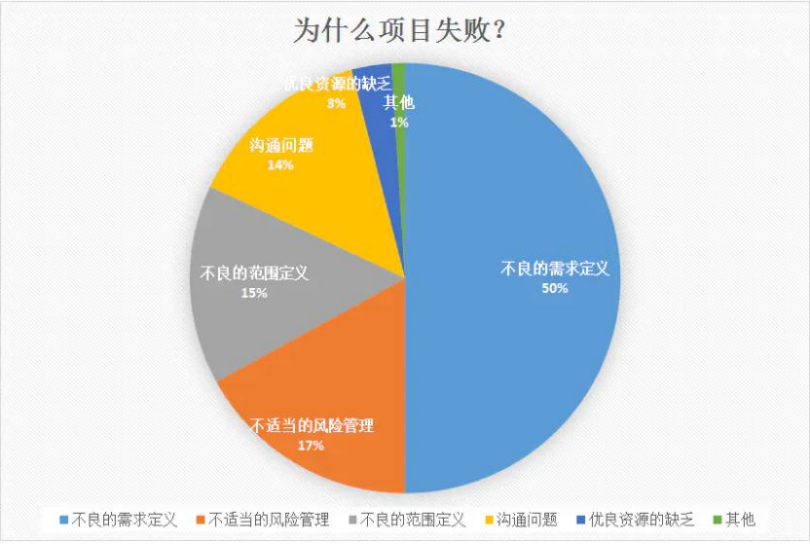
在软件项目开发中，一个系统所要实现的最终目标取决于需求，需求明确了客户对系统功能、性能等相关层面的详细需要。需求催生出购买意愿，产生消费的动机，是发生交易行为的基础，因此需求还是推动客户和公司业务快速发展的动力源，甚至决定了政企底层的管理模式和商业模式。

相较于以交付软件功能为目标的传统项目管理方式，敏捷管理对于产品成功的定义更侧重于满足用户的需求，通过实现如何更好满足用户的需求，将软件真正的价值交付给客户，从而获得在商业上的成功。因此越来越多的公司开始效仿这种行为去思考、分析、挖掘用户背后的真正需求。

举例：例如淘宝和京东在电商领域平分天下的局面下，拼多多创始人黄铮利用早年在做页游时对用户心理的拿捏，将游戏中的各种机制带入到拼多多的产品设计和日常运营当中，并通过对下沉市场用户需求的准确把握，成功实现了逆袭。抖音更是抓住人性共通这一特征，将本土经验成功复制到海外市场，才有了 TikTok 今天在海外的风光无限。这背后除了复杂深奥的技术外，更有着对用户需求，对人性心理等相关因素的深刻认识，而这些认知强化了软件的产品力，才让这些 APP 在一轮又一轮的竞争中，逐渐成为融入人们日常生活中的一部分。

2.1.2 需求的重要性

对需求的管理决定了一款软件的成败，以下图为例，如果关于需求的工作没有做好，项目就等于在一开始便失败了一半。因此，做好需求可以事半功倍，做不好需求很可能导致项目最后功亏一篑！



2.1.3 需求的特点

（1）多样性

追求便捷、追求文艺、追求经济价值、…

（2）复杂性

业务流程复杂，技术实现困难。

（3）隐蔽性

用户无法直接用语言清晰描述。

（4）差异性

角色不同，需求不一样，有可能还存在冲突。

举例：用户、商家、运营、细分角色

（5）变化性

用户的需求在升级，要求不断变化

举例：人找数据到数据找人

（6）矛盾性

隐私与便捷到底谁对谁错？推荐算法的是与非

明确写在合同上的——条款

用户开会说出来的——要求

用户没直接表达的——期望

2.2 需求的价值分类

2.2.1 用户价值分类

（1）业务闭环类价值

闭环：不依赖或者极少依赖外部输入，通过打造系统内部生态，完成价值的内循环。

举例：用户在电商买商品，留下浏览记录和下单记录，电商平台再通过大数据分析得出该用户的用户画像，根据用户喜好精准推送“猜你喜欢”，形成闭环，提升复购率。

举例：用户通过搜索引擎查找信息，留下的词条供搜索公司进行数据分析，提取上月价值，通过将词条竞价排名，帮助商家实现引流和变现。

举例：抖音、快手等直播平台，通过打造直播的电商的模式，开发平台自身流量的价值，通过打造电商闭环，对传统平台型（京东、淘宝）或内容型电商（微商、公众号）进行市场份额的挤压。

传统的内容电商场景以微信为流量入口，基础微信庞大的用户基础，打造内容电商，包括社群、公众号等，然而直播电商对平台型电商、私域电商等场景进行了差异化的竞争。

针对电商而言，技术上实现的任何酷炫且丰富的功能本质上都是为流量而存在的，流量服务于销售，销量最终才决定平台和商家的兴亡。因此，不见得最好的技术就有最好的效果，决定平台生死存亡的是产品和服务，而不是工具本身，技术要能找到符合自身价值的场景，且恰到好处的为业务服务，才能发挥出最大的价值。

（2）效用类价值

举例：便利性（打车、外卖）、安全性（保护隐私、保证交易安全）、耐用性（系统稳定性强，并发能力强）、经济性（省钱或省时）

（3）体验类价值

举例：身份权威（会员等级、会员权益、特权）、社会关系（人际关系连接、社区、圈子、论坛、贴吧）、隐私（私密空间：博客、微博、QQ 空间、云盘）、快乐愉悦（短视频、游戏）、回忆（相册、朋友圈…）

2.2.2 商业价值分类

（1） 经济效益：帮助用户促成交易、提升收入（O2O、电商）

（2） 数据效益：获取更多用户数据，帮助企业或政府做精细化运营（金融风控、精细化管理、精准营销）

2.2.3 社会价值分类

公益：帮助弱势群体就业（客服）、重点事件发生时的公共信息发布（微博）、帮贫困地区带货（电商直播）、帮助山区儿童获得教育机会（在线课堂）

2.3 敏捷管理下的需求管理

卡诺模型（Kano Model）

（1）魅力属性

让用户喜出望外的属性，不提供也不会降低用户的满意度；如果提供了用户满意度会大幅提升。

案例 1：微信作为一款即时通讯软件，提供语音短信、文字、视频、图片等核心功能外，还提供表情管理和摇一摇功能，满足了用户跟陌生人社交以及表达多样化的需求。

案例 2：支付宝作为一款实时支付软件，除了一般支付功能外，还提供记账功能，帮助用户记录日常开销的明细。

（2）期望属性

也叫线性属性，客户满意度与产品的属性呈线性关系。

举例：软件使用简单，用户不需要掌握复杂规则也能轻易上手、软件能根据用户喜好提供个性化内容、软件提供便利性省去大量时间、软件不卡运行流畅。

（3）无差异属性

用户不敏感的属性，无论提供不提供，用户的满意度都不会改变。

举例：用户不关心平台服务是自建服务器机房还是云服务，只要能保证服务的稳定运行稳定不崩溃不卡顿。

（4）必备属性

用户对产品的基本需求，如果不能提供，会导致用户满意度大幅降低，甚至无法接受。

举例：商城不能搜索商品、不能根据商品类型进行快速导航、不能退货等。

（5）反向属性

用户根本没有此类需求，提供的越多，用户越产生抵触情绪。

举例：各种强制广告弹窗。

总结：在企业真实经营中，时间和资源往往都是有限的，因此我们在做项目时应当优先满足产品的必备属性，若无法满足，用户则很有可能拒绝使用，直接导致项目失败。其次，满足用户的期望属性，能够直接提升用户的满意度；在资源有余力且不大面积超出项目范围的情况下，可以再去适当满足魅力属性，使产品尽可能的与竞品拉开差距；不必刻意花费资源去满足用户的无差异属性；警惕反向属性，通过认真调研跟严格评审尽力避免出现画蛇添足弄巧成拙的情况发生。

2.4 收集需求的常用方法

2.4.1 头脑风暴

项目组成员（产品、需求、项目经理、设计、开发）与甲方一起通过讨论集思广益，畅所欲言。举例：早期 QQ 软件的 QQ 秀就是内部员工向马化腾建议，最后经过大家讨论确定下来的，该功能帮助腾讯实现了最早期的流量转化，帮助企业盈利赚得了第一桶金。

2.4.2 访谈

访谈有经验的项目参与者、发起人、专家等，帮助识别和定义产品所需的功能。

举例：淘宝在创业初期非常注重中小商家和个人卖家的体验，马云和他的团队经常亲临用户一线去访问他们在使用系统时的感想，早期马云为了解决交易双方的信用问题下定决心打造支付宝，以平台信誉为担保，促使交易双方达成合作。

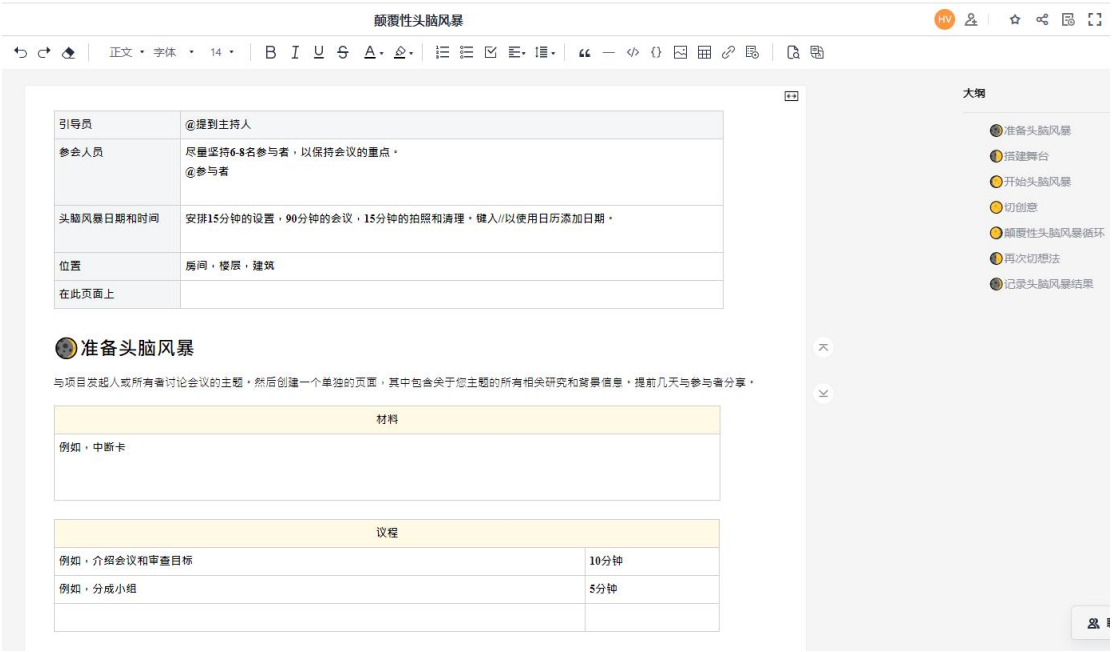
2.4.3 联合应用设计或开发（JAD）

召集甲方、业务专家、技术专家、以及其他相关方等，大家集中在一处，经由一系列的讨论和交流，收集用户需求和改进软件开发过程。 通过客户的持续参与，帮助客户充分了解项目并及时给出反馈，也有利于技术团队更深入的了解客户的真实需求。

2.4.4 标杠对照

将产品方案或已落地功能与其它同类型产品在功能上进行比较，以便识别出最佳实际，往往市面上已存在一段时期的产品都是经过用户筛选完成了市场检验的，因此与之形成对照，有助于形成改进意见，对照可以是外部的也可以是内部的。

华为云举例



2.5 需求管理计划

2.5.1 如何规划、跟踪、报告各种需求活动？

举例：规划需要多少个需求人员、多长时间、到用户的哪些部门，调研哪些内容、在什么时间内完成需求的采集，采集后向谁报告，通过怎样的形式来展现？文档还是原型？用户不满意的话怎么处理，多长时间内解决等。

2.5.2 针对需求的开发如何落地执行？

配置管理：针对用户要实现的项目目标，通过明确定义软件的功能及非功能性需求的各项具体参数，如：实现具体哪些功能点，系统响应速度为多久，最大支持并发数量有多少个，并在软件设计和开发过程中跟踪需求规格说明书中的这些关键参数，严格控制这些参数的变更，按照既定的配置执行开发计划，记录并报告各项参数的实现情况，最后通过测试和检查确认配置均得以实现。

管理对象

（1） 产品功能

（2） 项目基准

（3） 组织过程资产（项目过程中沉淀的技术知识、业务知识、经验、教训）

管理目标

完整性：确保在投标合同里、需求规格说明书、或其他书面文档中所定义的功能都得到了满足，没有漏项；

一致性：保证最终软件的功能、性能参数、都和配置计划保持一致；

可控性：保证软件开发的过程和质量可控；

追溯性：如果系统 bug 不能及时解决，可以随时回滚到历史可用版本。

管理步骤

第一步：定义产品配置

第二步：各种参数，控制变更

第三步：执行计划并通过记录来监督执行计划的情况
第四步：测试检查

2.5.3 需求能否发生变化

相比互联网企业的 2C 商业模式，因为传统 IT 企业的商业模式主要偏向的市场是 2B，业务主要是针对某一行业的专业领域的深入，所以通常采用较为传统的项目管理方式，即瀑布模型、增量模型、迭代模型，针对一些较为特殊的项目，也会采用增量和迭代一起使用的混合模型，这样做的好处在于一旦完成对需求的调研跟分析后，通过和客户对需求规格说明书上的内容进行确认和签字，在后面的软件设计和开发环节便不会发生较大的变化，等于为项目的成本时间等一系列要素设定了一条基准线，通过控制项目的不确定性，将项目风险降到最低。如果因为某些原因，如客户想修改需求，或者其他原因等，导致需求发生变更，双方必须严格遵循软件项目的变更流程，再经过 CCB 分析变更的合理性以及对基准的影响大小后，最终判断变更是否给予通过，由此可见，需求变更牵动着整个软件项目的各项基准，不是谁拍脑袋说变就能变！

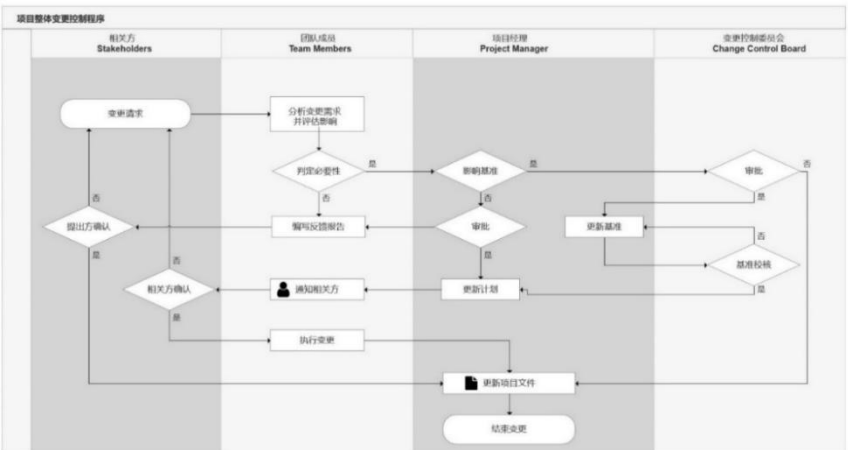
2.5.4 当需求发生变化如何管理需求变更？

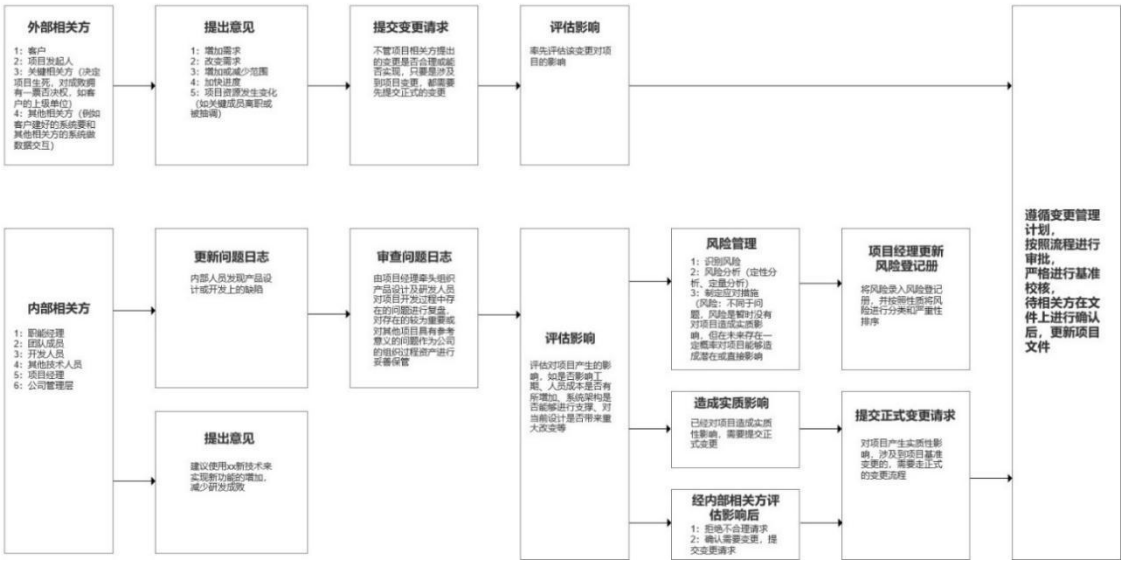
变更管理：如果需求发生变化，需要严格遵循并执行整体变更控制程序。变革管理可以看做属于配置管理的子系统，通过变更控制来保障配置管理中的各项参数可以得到有效的实施。

2.5.5 变更控制委员会（CCB）

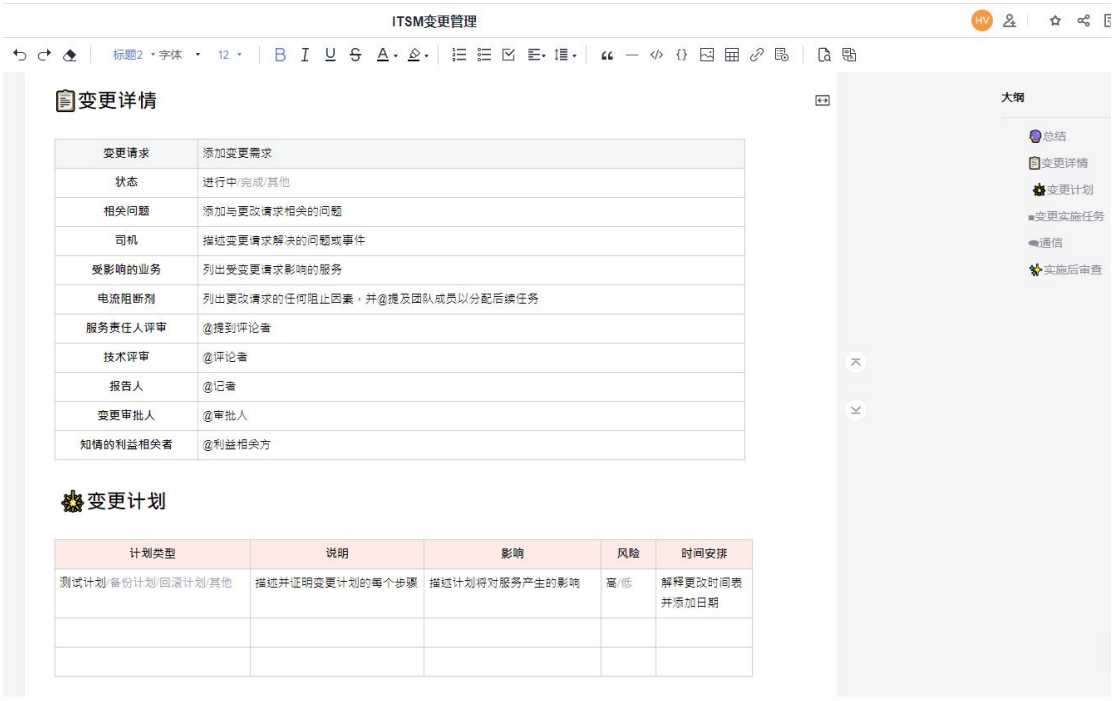
Change Control Board 是在项目立项后，由项目相关方组成的一个常设但是非固定的正式团体。配置管理和变更管理关乎项目的基准、产品最终实现的效果等，对项目成败起到至关重要的作用，几乎跨越整个项目管理的生命周期，因此 CCB 是在项目最终验收前的一个常设机构。但为了维护组织的公正性，团体内的成员可以是非固定的，内部的成员也是可以调换。因为变更对项目造成的影响实在是太大，所以并不是任意团体或个人都能随意决定，CCB 拥有对变更进行批准的唯一授权，通过评估影响来批准或拒绝变更请求，是一个受到项目各个相关方公认的正式组织。

2.5.6 变更控制程序

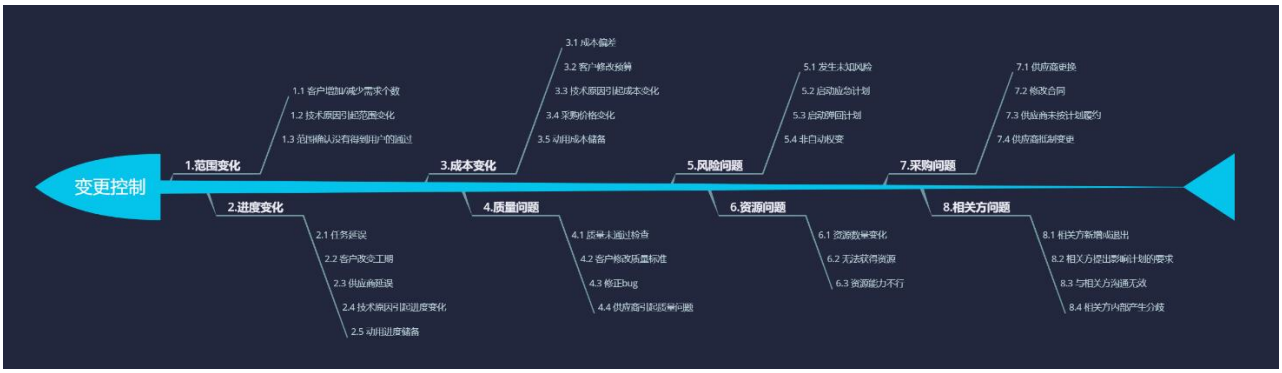




华为云举例

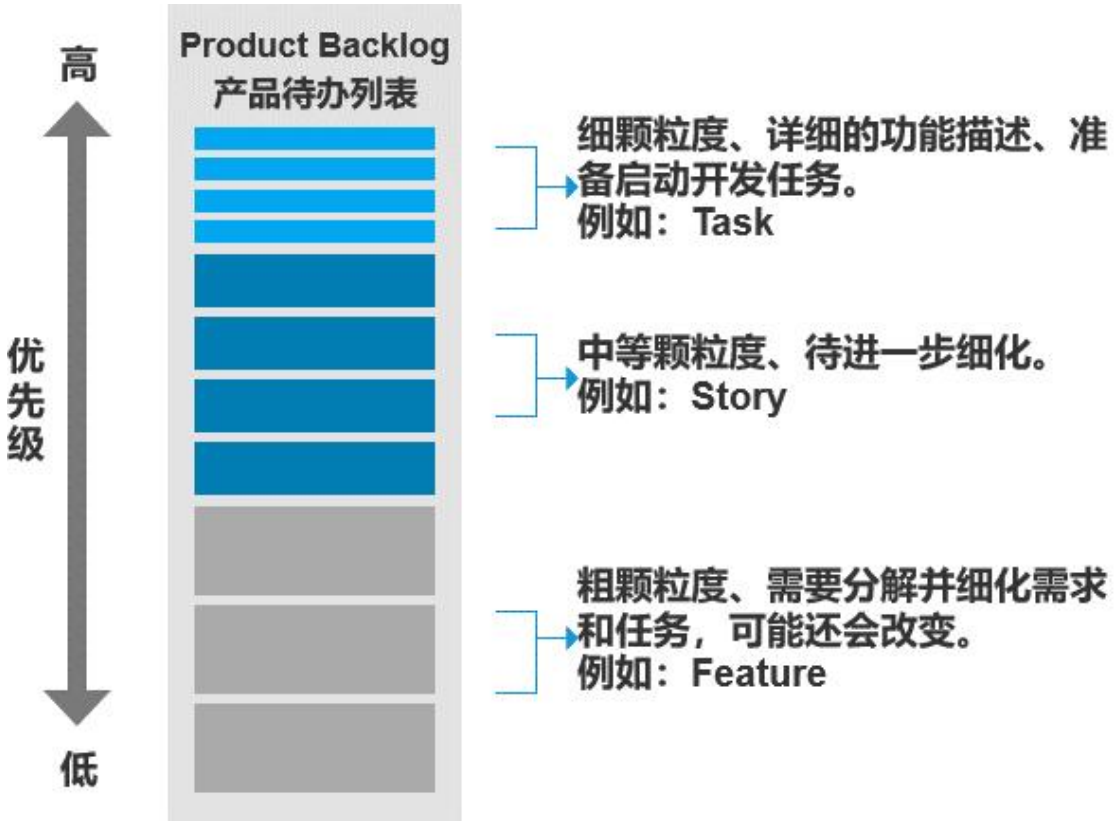


2.5.7 引发变更的常见原因



2.5.8 如何对需求进行优先级排序

产品待办事项列表 Product Backlog 的 Deep 特征
Detailed appropriately 详略适当的
Estimated 经过估算的
Emergent 涌现的（动态、灵活、随时可以增减和调整优先级）
Prioritized 按优先级排序的



详略适当的

优先级越高的待办事项描述应该越详细，需求明确、流程清晰、业务数据口径正确无误、产品功能经过原型评审且跟客户确认完毕，绝大部分工作不需要再去跟各方进行确认，开发人员拿到任务后可以直接进行开发工作。优先级越低、细节越少，不确定性越高，能让发开人员大致了解接下来工作方向即可。

经过估算的

产品待办列表中的任务条目是经过技术经理和开发人员粗略估算的，这些估算是粗粒度，通常是理想情况下的人天/人月。实际实施过程中，可能会受到各种因素影响，如客户的需求变更、客户的第三方技术供应商的配合进度缓慢等诸多外界因素。

涌现的

产品待办列表中的条目是动态变化的，随着对用户需求挖掘的深入，业务的扩展，它会随之不断演化，新识别的条目会被加入，现有的条目根据最新的用户

反馈不断调整、细化、甚至删除。条目的优先级也会被重新排列。

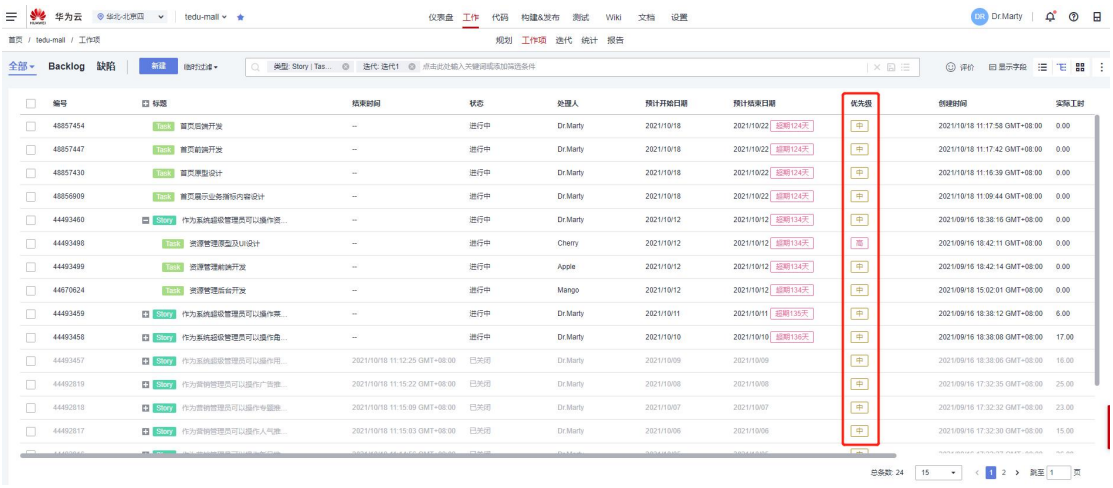
问题：列表中同样重要的任务先安排做哪个？

根据 WSJF 原则，采用“最短作业优先”的策略，WSJF = Cost of Delay/Duration，分母是这项活动的历时时长估算，分子是延期满足的代价。给每项活动计算 WSJF 分数，分数越高的越优先做。

华为云举例

Backlog 描述

Backlog 列表



2.6 需求管理工具

2.6.1 软件需求规格说明书

阶段：需求分析阶段

作用：需求规格说明书是需求分析人员（一般是企业产品经理），在进行完

市场调研并对用户的需求进行分析后所编写的文档。主要是帮助用户、设计、开发人员，明确系统建设内容，以及建设后所要达成的目标。除了例举客户当前的痛点以及论述产品开发的必要性外，最重要的是站在客户的业务需求、功能需求、以及非功能需求进行系统地分析。例如客户的业务场景、业务流程、业务规则、数据口径、与外部系统的交互关系、原型界面、接口规划、软件响应速度等。是用户与企业的技术合同说明，作为设计开发人员下一步进行设计和编码的基础，同样也是产品被市场或客户验收时的重要依据。需求规格说明书最终需要得到用户和企业双方的一致认可，需要用户、业务人员、技术人员都能看懂，一定是站在客户的角度，尽量从用户业务出发，用言简意赅的话语、清晰的图表等进行编写，尽量不要用过于专业的计算机领域的术语。

2.6.2 用户故事

阶段：需求分析阶段

作用：表述用户需求的固定语法格式。

作为一个<角色>，我想要<活动>，以便于<商业价值>。

通常产品和研发团队在识别用户需求时，按照上述固定语法格式将用户的需求表达出来，用于产品待办事项列表分析中。

华为云举例

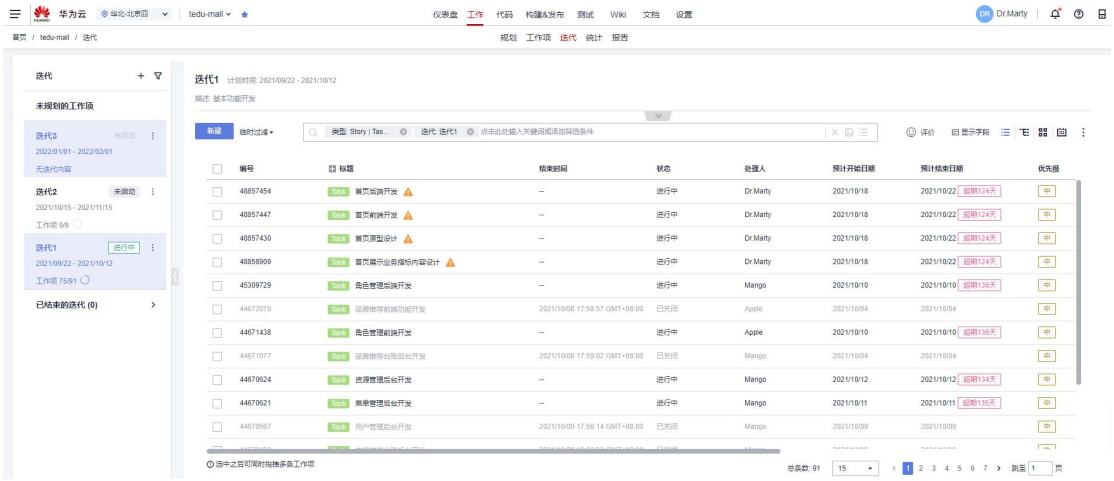
2.6.3 用户故事地图

阶段：需求规划阶段

作用：敏捷项目管理中的流行方法，将产品待办列表变成一张二维地图。）

1. 更容易看清系统规划的全貌
2. 为新功能筛选和划定优先级提供参考依据
3. 帮助规划迭代和增量式开发
4. 便于管理项目范围
5. 帮助管理者从多个维度思考项目规划（商业价值、业务流程、用户核心需求）

华为云举例



2.6.4 需求跟踪矩阵

阶段：全阶段（跨越需求分析、设计、开发、实施各个过程中的每个阶段）

作用：把产品需求从来源到能满足需求的可交付成功的一种表格。使用需求跟踪矩阵，能够把每个需求与业务模板或项目模板联系起来，有助于确保每个需求都具有商业价值。提供了在整个项目生命周期中跟踪需求的一种方法，有助于确保需求规格说明书中的每个需求在项目结束的时候都能被交付。最后，需求跟踪矩阵还为管理产品范围的变更提供了框架。

需求跟踪矩阵								
项目名称								
成本中心								
项目描述								
编号	关联编号	需求描述	业务需要、机会、目的、目标	项目目标	WBS可交付成果	产品设计	产品开发	测试用例
001	1.0							
	1.1							
	1.2							
	1.2.1							
002	2.0							
	2.1							
	2.1.1							
003	3.0							
	3.1							
	3.2							
004	4.0							
005	5.0							

项目编号：xxx-xxxxxxxx						项目名称：酷鲨商城运营管理平台							文件类型：需求跟踪矩阵							
用户需求										需求分析		产品设计		软件开发		软件测试		验收交付		
模块	子模块	功能点	需求编号	需求来源	详细描述	需求状态	变更标识	变更序号		名称	工作成果（WBS）	名称	工作成果（WBS）	名称	工作成果（WBS）	名称	工作成果（WBS）	名称	工作成果（WBS）	
功能需求	商品上架	添加分类	001	运营部门	提供后台类目管理	已完成	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
		添加属性	002		提供商品属性设置	已完成	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
		添加图片	003		提供商品图片添加	已测试	变更	1.1		需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
		添加详情	004		提供长图详情添加	进行中	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
	库存设置	设置图片	005		提供SKU图片设置	已批准	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
		设置库存	006		提供库存数量设置	已批准	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
		设置价格	007		提供SKU价格设置	已批准	原始			需求分析流程调研	需求规格说明书	原型绘制	产品原型	代码开发	功能实现	编写用例	测试用例	编写大纲	验收文件	
性能需求																				
安全需求																				
质量需求																				
运行环境																				

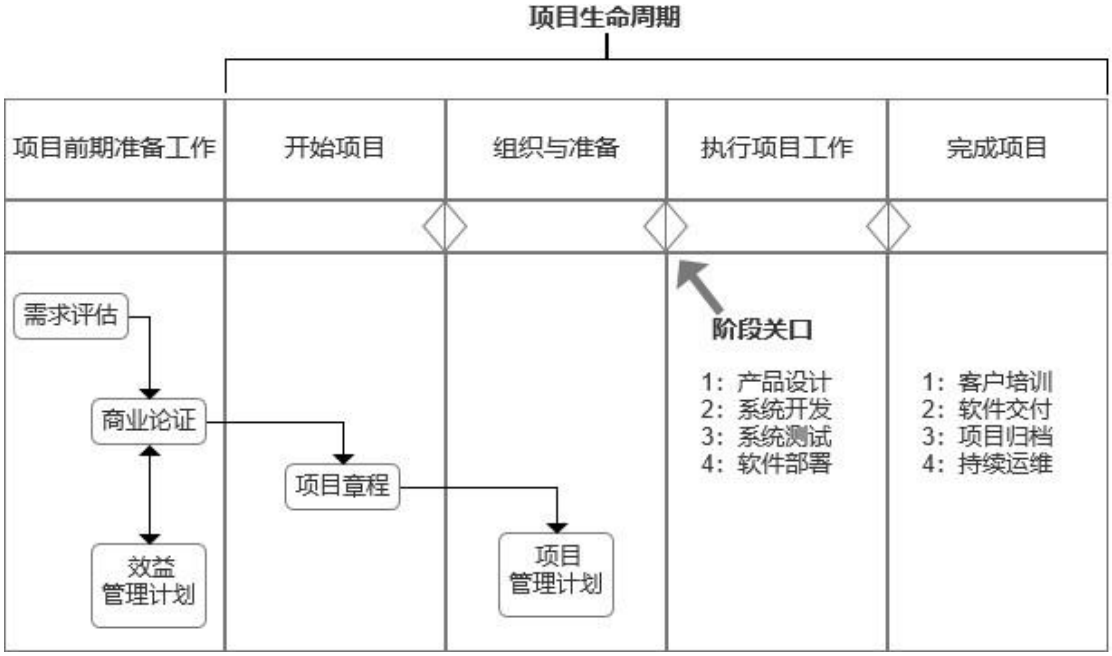
具体说明

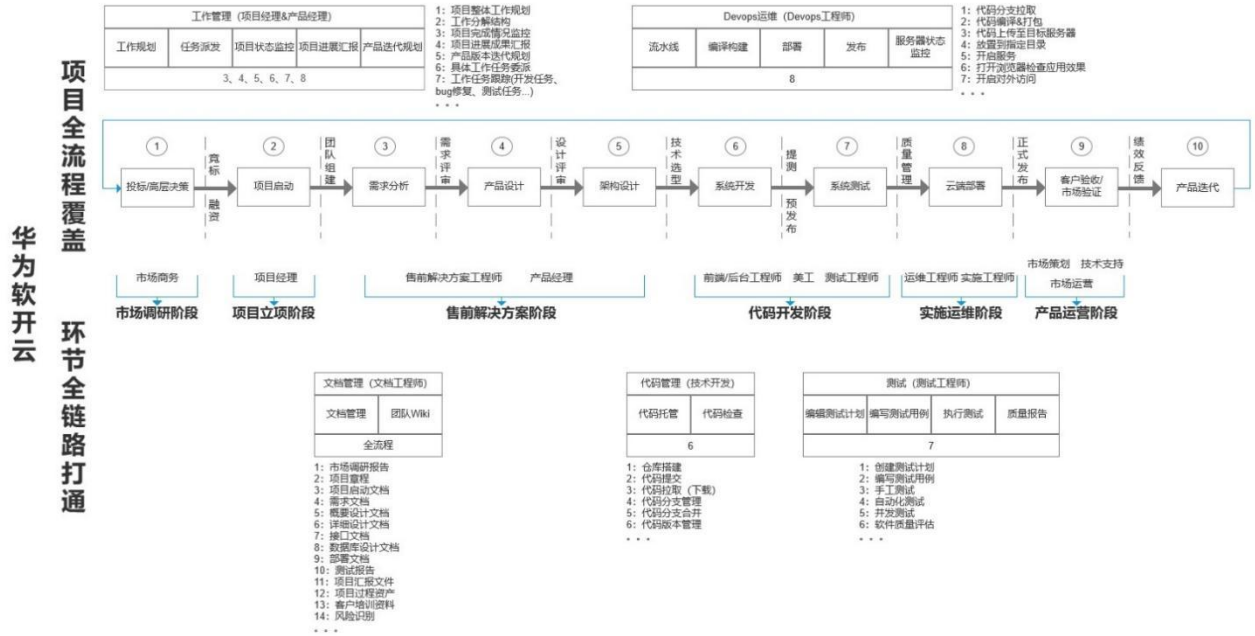
以电商为例，传统的线下零售模式基于选品、根据商品特征来锁定目标用户、再进行选址、最终通过宣传触达客户的四个过程。而在电商的业务场景下，底层的商业模式发生了本质的变化，从人找商品，转变为通过搜索或算法等技术手段在全网内精准触达目标群体，商家的策略重心也从选址变为了在特定节日下对平台流量的争夺，而如何为这些经营不同领域的商家，喜好各异的用户，制定出个性化的服务，并支撑如此庞大的流量，就是典型的互联网思维下的业务驱动技术，技术为场景服务的典型应用。

三、项目管理

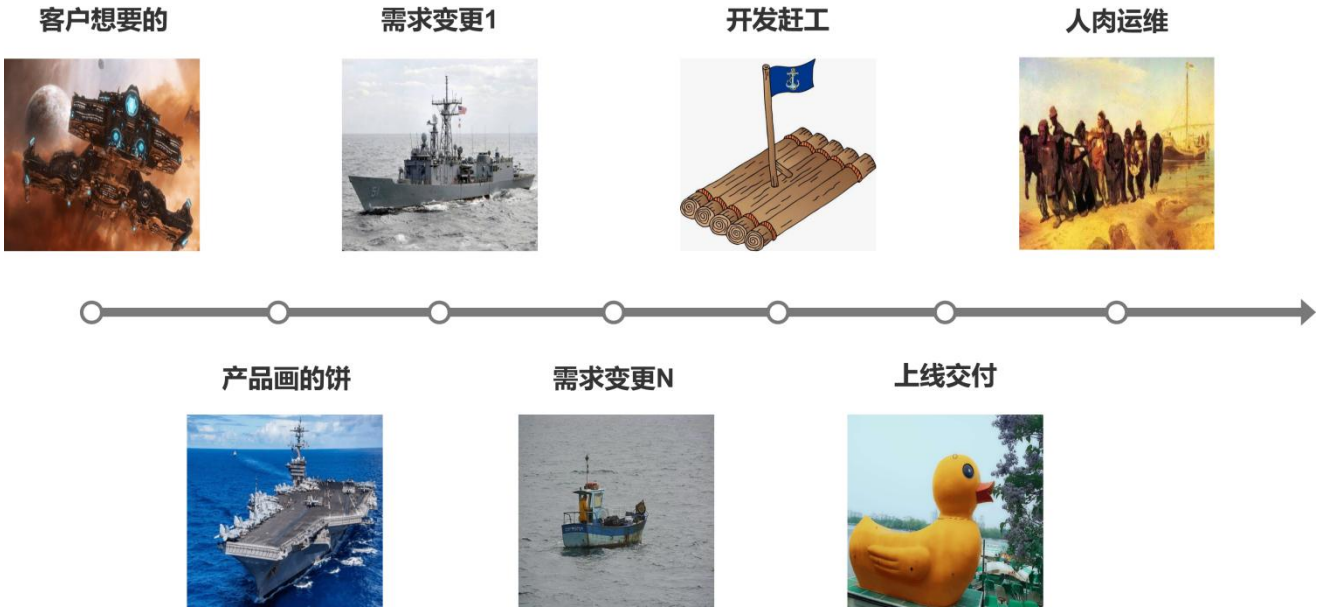
3.1 项目生命周期

3.1.1 通用项目周期





3.1.2 现实中的项目



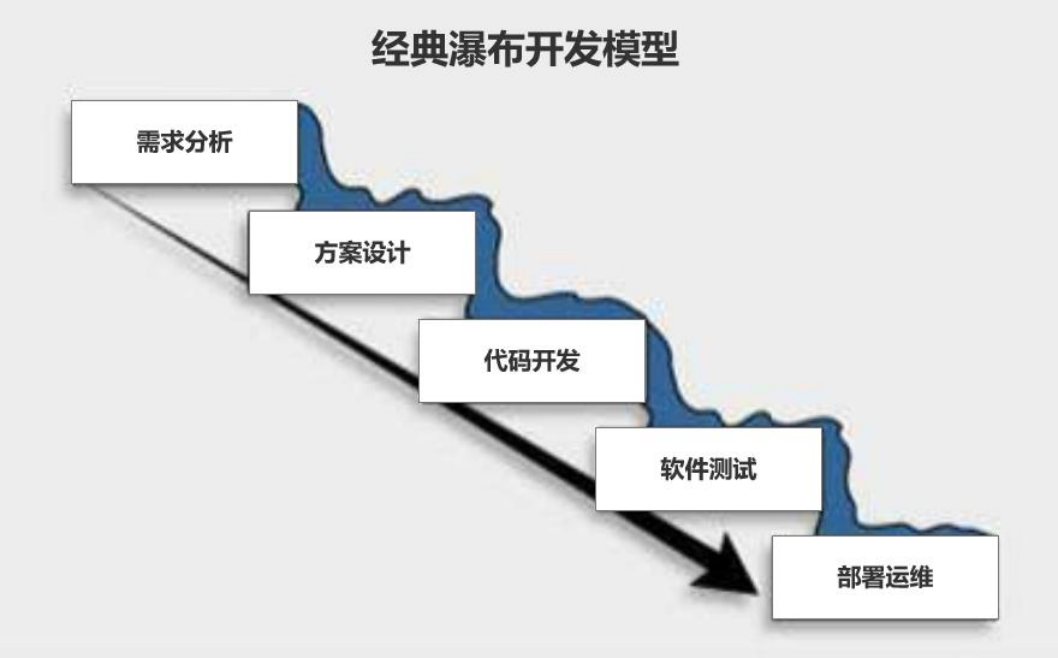
3.2 传统项目管理模型

3.2.1 瀑布模型

特点：瀑布模型是一种预测型生命周期，它的特点是在项目实施之前，通过事先制定严密且可控性强的详细计划，对项目的进度、成本、质量进行管理，给项目投资评估和精细管理奠定了基础。然而，预测型生命周期对变更很不友好，项目是一种系统性的工作，资源的投入跟项目进行的时间成明显的非线性关系，尤其是项目在进行到后期，变更将会产生极大的影响，代价大到几乎无法接受。

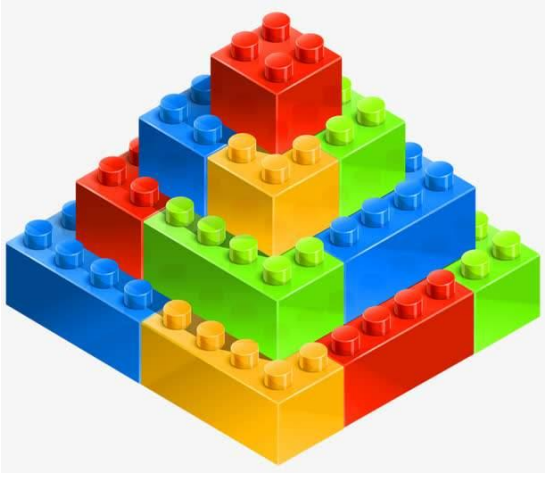
举例：在一款软件正式研发之前，已经确定要做几个系统，多少个模块，多少个

功能点，甚至接口的数量，数据总量和增量也都能大致评估出来，并根据这些信息分析得出项目所需投入的人力物力成本，以及预估工期长度，这样便能够在开始之前就预测出项目在交付时候的模样，不光如此，还能预测到未来在实施过程中每个阶段的进展，例如项目进行了两个月，需求调研工作进展的如何，产品设计进行到哪个地步，系统技术选型和脚手架是否确定并搭建完毕，数据库设计的雏形是否实现等，之后项目投入的成本将会花掉多少钱，半年后项目会进展到什么程度，软件能完成具体哪些模块，这一切都在计划之中，在项目经理的严格掌控之下，彷佛整个项目都是按照当初计划那样自己一点点长出来得一样。



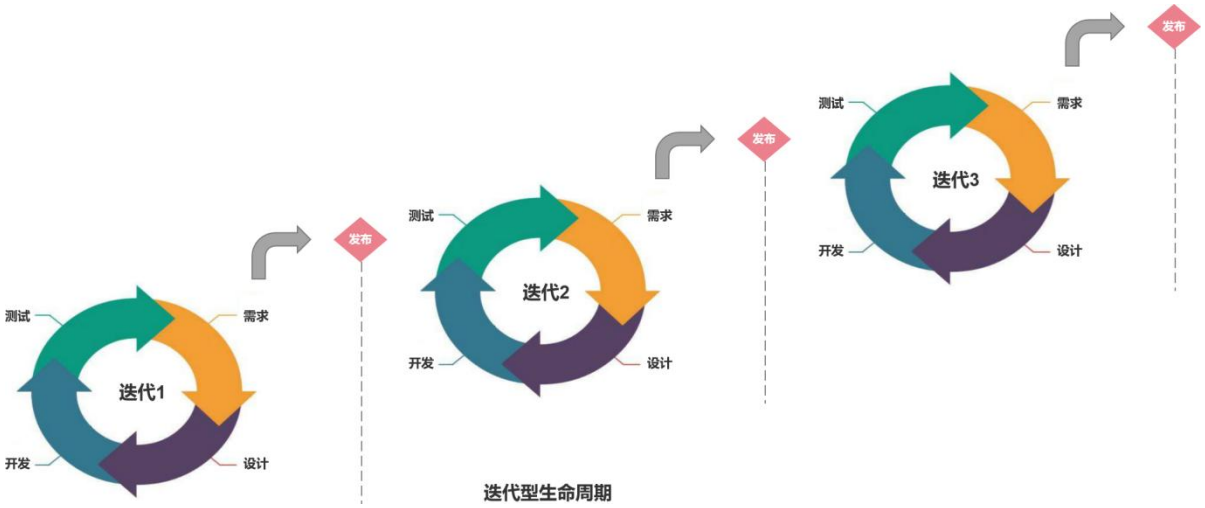
3.2.2 增量模型

特点：增量指的是先交付一部分，然后每次再交付一部分，产品就像搭积木一样，一块一块搭建而成。这样做的局限之处在于，核心业务流程可能需要几个业务模块的共同支撑，因此往往必须要等到交付完毕之后，业务才能够在线上真正的跑起来。



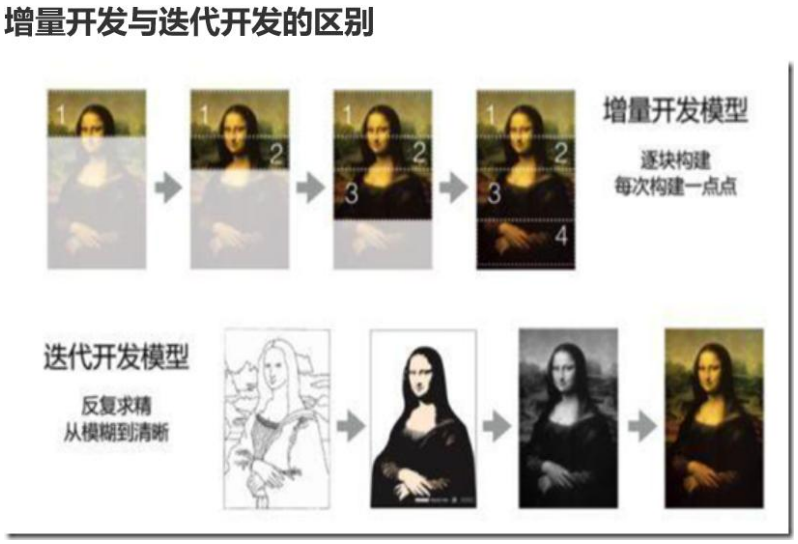
3.2.3 迭代模型

特点：迭代指的是多次循环，例如在软件开发中，按照版本发布，每一个版本内部就是一个小的瀑布开发，经历需求分析—设计—开发—测试—发布周期，下一个迭代再此基础上重复这些步骤对软件进行优化升级，再发布新的版本。



迭代型生命周期

3.2.4 增量与迭代差异

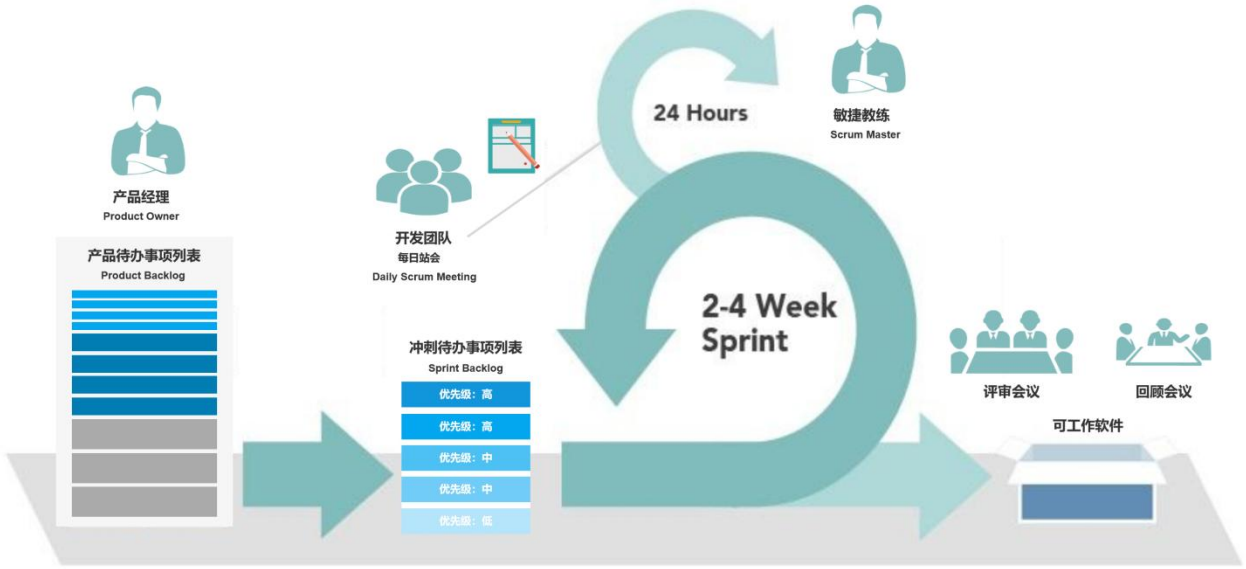


如图：增量开发的特点是每次构建一部分，逐块构建。以上图为例，把画分成借给部分，每次先完成一部分；而迭代开发的特点就是从模糊到清晰，反复雕刻求精，通过先勾勒处草图，确认方向正确无误后，再一步步进行细化工作。

3.3 敏捷项目管理模型

3.3.1 敏捷模型

特点：



Scrum 管理框架

3355 原则

（1）3 种角色

PO、SM、DevTeam（产品经理、敏捷教练、开发团队）

（2）3 项工作：

产品待办事项列表、冲刺待办事项列表、增量

（3）5 种事件：

冲刺规划会议、冲刺、每日站会、冲刺评审会、冲刺回顾会

（4）5 种价值观：

沟通、简介、反馈、勇气、尊重

3.3.2 PMF（Product-market fit）

1. 什么是 PMF

PMF 直译的含义是产品与市场的契合度，即产品在发布后首先要去验证产品在市场中的价值，通过从一小部分用户当中获取对产品使用的反馈，再采用小步快跑，快速迭代的方式让企业以最小的成本对产品进行改进，从而完成对市场中目标用户的争夺。

2. 为什么要通过小步快跑，快速迭代的方式进行产品升级？

（1）用户往往不清楚自己要什么，需求是模糊的，用户很难用语言清晰且完整的表达自己对产品功能的具体要求，且用户的需求还会经常发生变动，很难用一步到位的方式满足用户对产品的所有期望。

（2）软件开发过程往往存在着需求的蔓延，需求的改变等多种状况，为产品最终落地带来了各种各样的问题及风险，通过一步到位的方式意味着必须严格遵循瀑布模型，虽然能保证软件的交付，但不能保证用户对产品的满意度，软件的价值可能无法达到预期效果。此外，瀑布模型一旦完成需求范围的确定，功能设计

的评审，架构的选型等，后期就很难有较大的改动，因为工程量大，所需投入的资源多，不允许企业有太多的试错机会与试错成本。而小步快跑，快速迭代相比于传统的商业思维，这种模式充分体现了互联网企业的优势，互联网思维是拥抱市场变化的，为了适应用户需求的快速变化，挖掘新的市场机会，互联网公司往往采用敏捷管理的方式，通过“大胆尝试，小心求证，来让市场给出产品的最终去向的答案”。

3. 如何科学试错？

首先我们要明确，小步快跑，快速迭代绝不意味着盲目试错，当你对产品和用户只有一个模糊概念时，如果就开始盲目试错，结果付出了大量的成本和精力，却没得到半点收获。因此在你进行试错之前，需要先问自己如下几个问题。

（1） 你对产品的定位是什么，能为用户解决什么问题？

（2） 相较市场中的竞品有哪些竞争力？

（3） 产品的独特性在哪里？

（4） 产品的劣势是什么？

（5） 如何掩盖和缩小劣势？

（6） 你的目标用户是哪类人群？

（7） 他们在哪些地方能找到，最集中的渠道都有哪些？

（8） 获取一个用户的成本是多少？

（9） 你能想到多少种吸引新用户的方式？

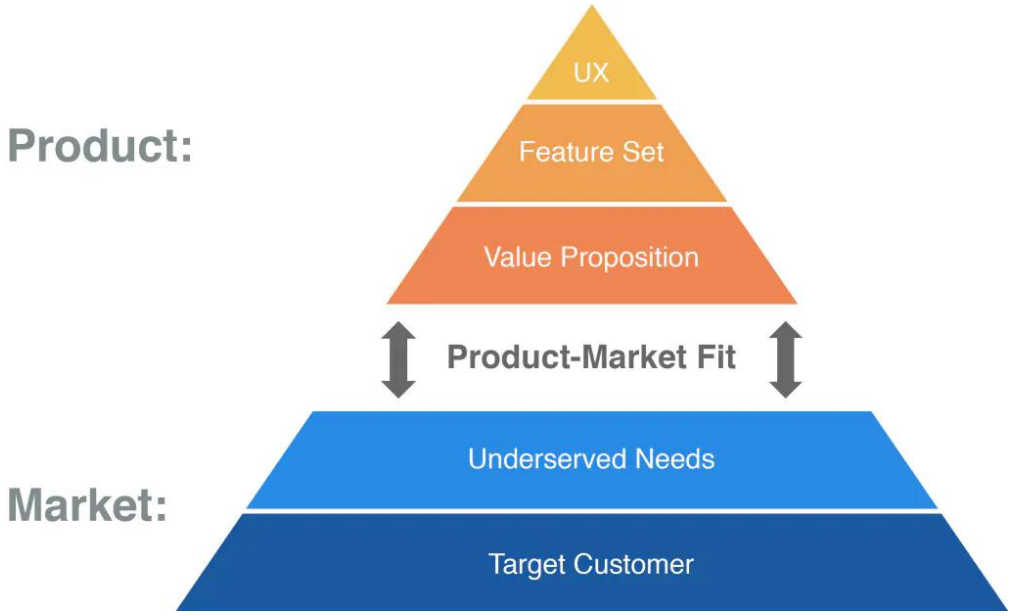
（10）有没有分析过做的好的案例及差的案例，以及它们产生的原因是什么？

由此可见，挖掘和定义需求的过程其实就是一个不断验证假设的过程，团队和个人在试错中学习与成长，逐步逼近直至发现问题的本质并找到与之相对应的解决方案，即产品功能、商业模式、企业价值观找到与市场最佳契合点的过程。

4. 针对上述问题该如何解决

（1） 对照市面上成功的产品进行参考，从简单临摹再到精益创新
（2） 进入到用户群体当中进行调研，和用户当面聊，看用户如何去使用
（3） 通过调查问卷的方式去采集用户反馈
（4） 通过技术手段分析用户的行为数据
（5） 聘请专家进行讲座或者拜访相关从业者

5. 什么是 PMF 金字塔模型



PMF 金字塔定义了五个层级，从下到上，这 5 层依次是：你的目标用户，你用户未被满足的需求，你的价值主张，产品的功能集，你的用户体验（UX）。

6. PMF 金字塔模型给互联网产品开发流程带来哪些启示

基于金字塔模型，可以根据层级的优先级来搭建一条易于产品迭代的开发流程，该流程可以通过假设和测试来不断修正产品的功能，从而逐步提高产品和市场的匹配度，该流程可以大致分为以下六个步骤。

（1）确定目标用户

（2）发现市场中用户尚未被满足的需求

（3）定义你的价值主张

例如：阿里让天下没有难做的生意、出门可以不用带现金，京东给用户最好的物流体验与产品品质保障、字节跳动的软件连猴子都会使用等…

（4）设计最小可行产品（MVP）的功能集

（5）制作产品的 MVP 原型

（6）完成 MVP 的客户测试

3.3.3. MVP（Minimum Viable Product）

（1）什么是 MVP

MVP 全称是（Minimum Viable Product）最小可交付价值，简单来说先实现产品的最核心功能，满足用户的最主要需求。

（2）为什么要采用 MVP

商业具有极大的不确定性，其不确定性主要表现在市场的需求是不断变化的，商业模式的日新月异，与技术不断的发展，所有要素都是一组动态的、待验证的假设。因此，如果一个环境中所发生的事件都高度随机的，给商业上的预测工作带来极大的困难，通过大手笔打造一个确定的产品并一举获得成功的可能性大大的降低，给商业带来极大的风险。因此，需要精通市场和商业规律的业务人员，基于市场调研与需求分析，让开发团队在初期通过交付一个最小化可行产品来尽快获取用户反馈，并基于此产品上，持续集成快速迭代，直到产品达到一个用户

满意度相对稳定的阶段。MVP 对企业来说可以快速验证对市场的假设，通过快速试错的方式，来达到与市场的快速匹配，其理念主要来源于项目管理中的敏捷开发。

7. 如何执行 MVP
（1）识别产品的关键价值，结合成本与风险对众多需求进行优先级排序。（产品功能待办事项表）
（2）采用少即是多的原则，Do less，给产品功能做减法。
（3）与需求的提出方进行沟通。 例如：谁的诉求、为什么客户需要、需求是如何产生的、应用的场景在哪、客户目前的实际情况是什么、该功能能够为客户带来什么价值、为什么提出需求的人觉得该功能高大上，为什么该功能非常必要、上线后产品后期怎样运营、未来该如何扩展、产品的投资回报率多少、投资回收期多久、市场风险多大。

工具：需求跟踪矩阵、场景故事地图、冲刺计划安排、预算方案、风险管理列表…

8. 制定一套可实施的业务指标
C 端：以电商为例
（1） 设置用户增长率
（2） 设置商品销售总额（GMV）
（3） 设置留存率
（4） 设置产品付费转化率
（5） 设置月度流失率
（6） 设置月度毛利
（7） 设置用户获取成本的回本时间<X 个月

B 端：以政务系统为例
（1） 减少员工每日处理工单的时间
（2） 降低整体投诉率
（3） 提升报表的准确率
（4） 提升人或物的查获率
（5） 提升事件的响应速度
（6） 提升数据统计效率
（7） 提升流程透明度

总结：C 端产品注重用户流量的转化和商业价值的变现，而 B 端更侧重的是提高政企的业务办理效率。

3.3.x 失败的敏捷

1. 没有需求或原型等相关设计物料，专业知识的交流全靠某些角色口喷，技术人员凭空开发。
2. Backlog 中的条目全是粗粒度，业务全靠猜，开发只能跟着感觉走。
3. 需求没有先后、不经过任何辨识和评审全盘接受。
4. 需求随意变更，客户想怎么做就怎么做，天马行空不切实际。

5. 需求或管理人员沦为项目传声筒，没有任何有效信息的价值被提炼的过程。
6. 加班时长成为团队正在实施敏捷的象征。
7. 系统版本迭代没有任何规划和能够落地执行的可行方案。
8. 产品跳跃式迭代，产品并非经过业务和技术的沉淀从而逐步迭代，也不是市场验证的结果，却期望产品能够一步到位。
9. 版本的迭代不是真实需求决定，而是少数人拍脑袋的想法。
10. 团队对项目的承载力以及所能够承担的项目风险并非经过严谨的商业论证以及对资源和能力的评估，而是少数人拍胸脯保证的结果。
11. 团队所实施的管理手段只剩下领导拍桌子。
12. 不断有团队成员在敏捷实施的过程中拍屁股走人。
13. 产品成功不以能为真正使用的客户交付多少价值为导向，而是为满足少数某些内部或外部相关方的自嗨。
14. 资源的投入不是根据项目各阶段的实际需要，而是不假思索的盲目投入，或是按照会哭的孩子有奶吃，老板小姨子先上路等优先原则。
15. 站会对项目实际推进起不到任何帮助，无法解决任何实际问题。
16. 没有角色进行阶段性复盘沉淀经验教训，团队无法从上个项目或迭代中积累知识跟经验，每个项目的建设都是从头做起。
17. 工作群和会议的数量远超它应有的规模，且绝大多数对项目实际进展起不到任何帮助。
18. 团队成员相互抱怨，敏捷教练对此熟视无睹，无法进行有效的干预。
19. “敏捷教练”无法为团队创造一个不受打扰的环境，不能够屏蔽外界的无效沟通或垃圾信息对团队成员的侵入。
20. “敏捷教练”把压力无条件传导至团队，或其本身的存在就是压力源。
21. “敏捷教练”不能帮助团队实现目标，仅仅是维持一种上下级关系，成为项目过程中产生或传递无价值信息的“肉喇叭”。
22. “敏捷教练”对业务或者技术一无所知，或仅了解名词，无法对产品设计或技术实施过程中可能出现的风险和问题提前进行预警跟确认。
23. “敏捷教练”没人任何激励团队成员的方式，也没有任何阻碍团队负面情绪蔓延的手段，加速项目走向崩溃。
24. Backlog 中的条目数量远超团队承载力且还在不断有新的条目加入，却没有任何旧的条目被调整或删除。
25. 团队成员对敏捷一无所知，对产品没有自己的理解，工作方式仍旧采用传统的被动接受的方式开展。
26. 团队成员不按照冲刺列表里的条目进行开发，而是凭借自己的喜好修改产品设计或调整开发的优先级。
27. 团队成员认为需求和设计仅仅是产品的工作，和自己没有半点关系。
28. 团队成员在开发过程中不考虑产品今后的迭代，对代码成果不负责任。
29. 团队成员不参与需求和设计的评审，不能站在各自的立场对产品进行思考从而提出任何有价值的建议。
30. 团队成员认为敏捷就是不假思索的变更，不写文档就是敏捷。
31. 组织没有为敏捷团队提供任何必须的资源或培训。
32. 组织不理解敏捷，没有从公司组织架构和人员设置层面进行相应的调整，而是仅仅期望下面团队的自主改变。
33. 组织不重视敏捷，有路径依赖的惯性，担心会对现有组织架构产生破坏，

因此无法从根本原因上解决问题

34. 组织对敏捷实施盲目乐观，认为只要招聘到适当的人选便能够马上产生效果。
35. 组织不重视成功经验的分享与方法论的推广。
36. 组织认为敏捷就是新瓶装旧酒，引入新的管理工具就是实施了敏捷。
37. 组织委派远离一线经验的人员充当敏捷教练。
38. 组织把敏捷实施效果差的原因归咎于个人能力。
39. 组织未能打造自己的企业文化或良好的生态环境，内部缺少变量，或纯粹依赖外部力量的介入。
40. 组织不重视人与人之间有温度的连接，此外，不同部门间由于在业务上存在不小的差异，加上企业又缺少复合型人才能够对不同业务线进行有效地统筹管理，导致管理上对部门墙的产生没有能力解决。

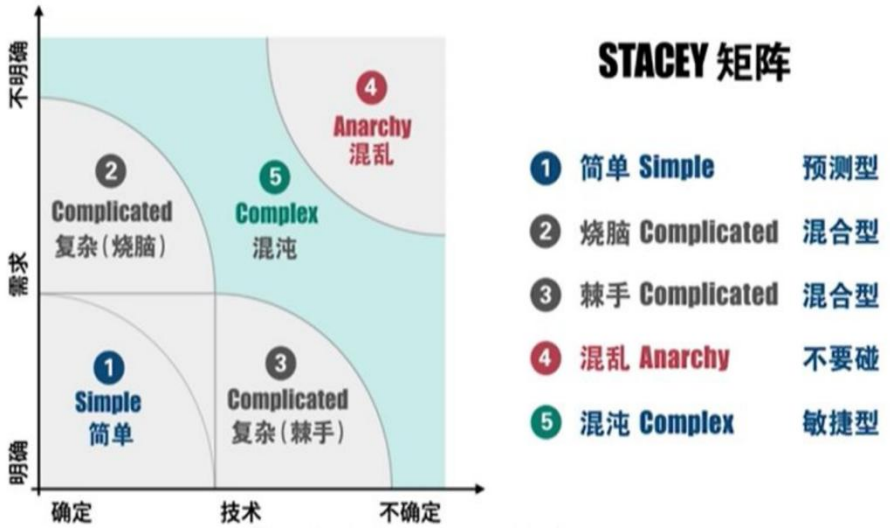
3.4 敏捷环境搭建

3.4.1 敏捷工具选择

（1） 选择适用的生命周期（Stacey 矩阵）

不同的项目场景应该采用不同的项目管理模型。

1996 年拉尔夫 D 斯塔西 提出第一个方法，帮助我们做项目判断应该采取哪种开发方式。



1 区

需求明确，技术方案也确定，这种就属于简单项目。

因为需求明确，实现手段非常清晰，便能够提前计划好方案，因此预测型生命周期最为合适。

2 区

技术很确定，需求却不怎么明确，这种是软件项目中非常常见的一种情况，客户不知道自己到底想要什么，问题不在于技术团队能提供什么样的能力，关键在于客户到底要什么，只有在提供一些现成产品供客户可选的时候，用户才能够通过他们不想要什么渐渐让产品团队勾勒出他们想要什么，这类项目就属于复杂项目中的烧脑型。

针对这种情况，建议采用混合型生命周期，例如在产品设计阶段，在原型开发时采用敏捷模式，待明确需求验证技术可行性后，在开发阶段采用瀑布模型。也可以采用逐块构建，通过增量交付的模式对项目进行推进，从而降低推倒重来的风险。

3 区

需求很明确，技术却不确定如何实现。这类项目属于复杂项目中的棘手型。例如智能翻译软件的需求很明确，但是实现的效果并不理想，借由人工智能技术的崛起，技术上日渐趋于成熟，但还存在一定困难。

这类项目建议采用混合型。概念阶段采用敏捷，软件开发阶段采用迭代。

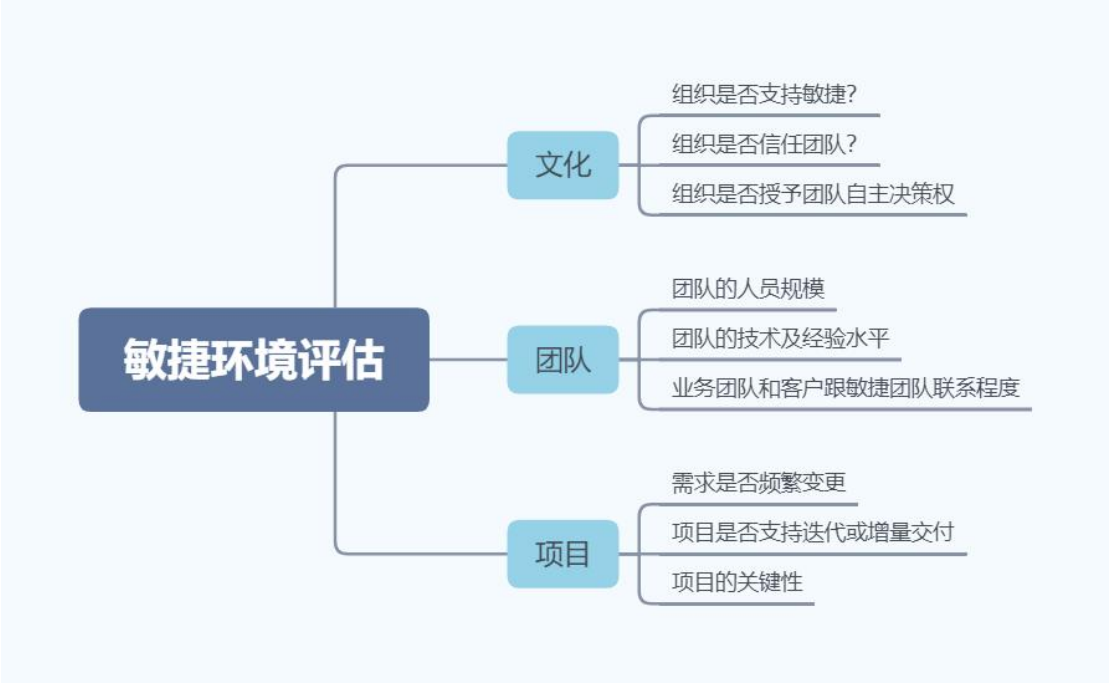
4 区

不但需求不明确，技术实现方式也不清楚，这种项目处于混乱状态，项目失败的风险非常高，若不是非做不可，建议远离！

5 区

需求的挖掘持续进行中，技术手段也需要再不断地探索，这种项目就属于混沌型，建议采用敏捷开发，小步快跑，快速迭代，通过科学试错，不断接近目标，拥抱变化，灵活调整。

（2）雷达图



敏捷环境评估主要从三个维度进行分析。第一：环境维度。企业文化是落地敏捷的关键因素，很多互联网公司能够很好地实施敏捷的主要原因，除了时代给予的发展机会外，最重要的就是企业从创立到发展过程中渐渐确定了围绕以包容、开放、平等、自由为原则的企业文化。此外，拥有这类文化氛围的企业，也大多采用扁平式的组织架构来减少官僚主义和信息差异对团队执行力造成的影响。如果企业是个层级森严、部门分工和业绩考核标准明确，在进行敏捷落地时，就会导致由职能部门的部门墙产生的沟通协作困难，各个部门更专注于自身资源的可用性与部门的短期业绩目标，而很有可能忽视客户的核心利益，项目整体目标等关键因素，因此不利于以矩阵型项目组织的形式搭建敏捷团队。这种情况也会导致组织从高层到业务部门的负责人也很难给予敏捷团队足够的信任，然而信任又是任何一个团队的基石，尤其在敏捷团队需要高度自主决策权的情况下，任何外界过多的干涉与组织内部的矛盾都会极大降低敏捷团队的战斗力。综上所述，企业若没有与敏捷相适应的企业文化则很有可能导致结构性矛盾从而影响敏捷落地执行的效果。

敏捷的成败还和团队自身息息相关，按照云计算巨头亚马逊的成功经验来看，一个敏捷团队的规模采用团队成员在加班时点两张披萨够分的原则，由此可见，敏捷团队的规模不应过大，应采取小而精的组织形式。此外，团队各个类型成员的能力也至关重要，经验不足、能力水平不够的成员会成为团队在进行冲刺时极大的短板。团队对客户需求、业务知识、产品目标的理解程度也同样决定了产品的最终效果，如果团队没有与业务足够强的联系，就会导致交付成果与客户期望的巨大偏差。

项目是否真的需要采用敏捷进行管理还需要看碟下菜，首先要判断项目需求是否频繁变化，如果需求很明确，则采用预测型模型更好。还要考虑项目本身和客户是否支持迭代或增量交付。此外，敏捷管理采用 MVP 原则，若产品或服务非

常零散，缺少关键性，则很难去框定最小可交付价值的范围。若项目本身存在以上这些情况，则不适合采用敏捷的方式进行管理。

3.4.2 敏捷环境评估

特征				
方法	需求	活动	交付	目标
预测型	固定	整个项目仅执行一次	一次交付	管理成本
迭代型	动态	反复执行直至修正	一次交付	解决方案的正确性
增量型	动态	对给定增量执行一次	频繁更小规模交付	速度
敏捷型	动态	反复执行直至修正	频繁小规模交付	交付客户价值

3.4.3 敏捷团队搭建

（1）人才选型

I 型人才：深耕某一领域，对该领域有丰富的业务或技术经验，但缺乏其他知识领域的广度。

T 型人才：既有较深的专业知识，又有广泛的知识面，体现出自身一专多能的特征，拥有良好的合作技能。

π 型人才：技术背景雄厚、精通业务知识、拥有广阔的产品视野、敏锐的商业嗅觉、快速适应迭代的能力，这类人才不但自身素质过硬，还拥有优秀的领导能力！

敏捷团队是跨职能部门的，团队的终极目标是打造一支由 π 型人才站在一定高度指挥战略方向并对产品进行指导和把关，由 T 型人才打通内部和外部各个方向的关键脉络，并带领及配合 I 型人才实现关键问题的攻坚。由此可见，敏捷团队需要每一个成员都具备成为专才或通才的潜质，接受变化不仅仅是体现在需求上，更是对个人成长的一种要求！

（2）设立团队工作场所

团队需要一个场所让他们能够一起工作，从而了解整个团队的工作状态，根据团队或其他成员的具体工作情况，成员之间能够及时沟通协作，以便于提高团队的整体工作效率。

集中办公：团队成员集中在一处，方便进行面对面沟通。

分散式团队：“鱼缸窗口”、“远程结对”；

鱼缸窗口：通过在团队分布的各个地点之间建立长期视频会议链接，创建一个鱼缸窗口，工作开始时，打开链接，工作结束时，关闭链接。

远程结对：通过使用虚拟会议工具来共享屏幕，包括语音和视频链接，建立远程结对。

3.5 敏捷落地实践

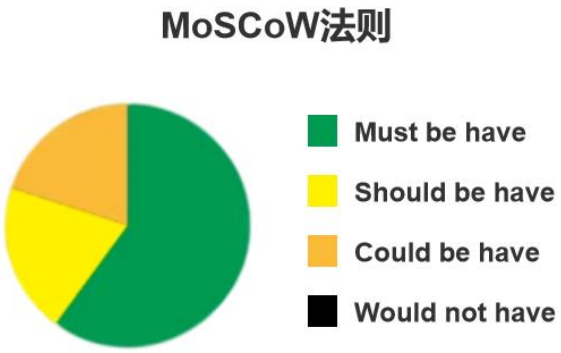
3.5.1 定义用户

通过采集用户信息，使用大数据技术堆用户进行分析建模，得出用户画像，对用户进行标签化管理。
人口属性、用户偏好、消费特征、地理位置、用户行为、设备信息…

3.5.2 编写用户故事

编写用户故事
卡片+会话+确认
（1）通用格式：作为<用户角色>，我想要<结果>，以便于<目的>
（2）3C 原则
卡片-card、会话-conversation、确认-conformation
（3）INVEST 原则
独立的-independent、可沟通的-negotiable、有价值的-valuable、可估计的-estimable、小型的-small、可测试的-testable

3.5.3 排列优先级



MoSCoW 法则
Must have — 这次迭代必须有的功能
Should have — 这次开发应该会有的功能
Could have — 这次开发可以有的功能
Would not have — 这次开发一定不能用的功能

3.5.4 用户故事估算

故事点估算
三点估算、类比估算、计划扑克

3.5.5 发布计划

用户故事地图

3.5.6 验收测试

AC（Accept Criteria）

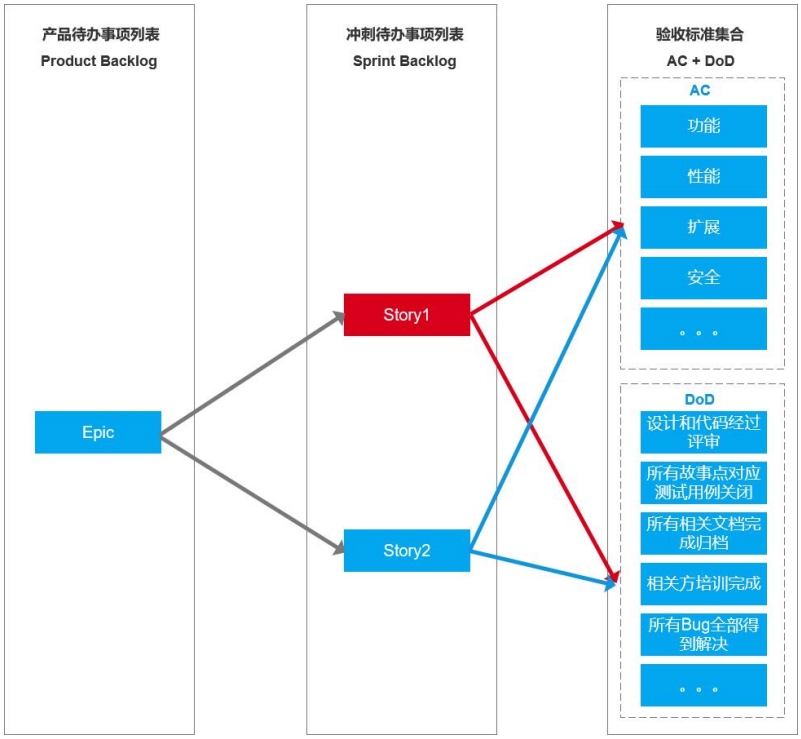
聚焦客户需求，关注外部质量、做正确的事情、与用户故事卡片搭配使用。

解释：用户对产品是否满意、功能及非功能需求等是否均满足。

DoD（Definition of Done）

聚焦企业工序，关注内部质量、采用正确的方法做事、内部达成对于完成的一致性。

解释：内部是否严格按照规章流程进行设计、开发、测试、部署，是否遗留技术债、产品功能是否模块化，能否通过技术移植快速支撑其它产品线的搭建等。



3.6　混合型生命周期

常见的混合型生命周期通常指敏捷和瀑布开发进行混合。例如在软件项目开发中，在概念阶段可以采用敏捷方法去确认用户的真实需求，验证技术方案；正式开发阶段，则采用瀑布开发加强计划性和对项目成本、进度、质量的可控性。