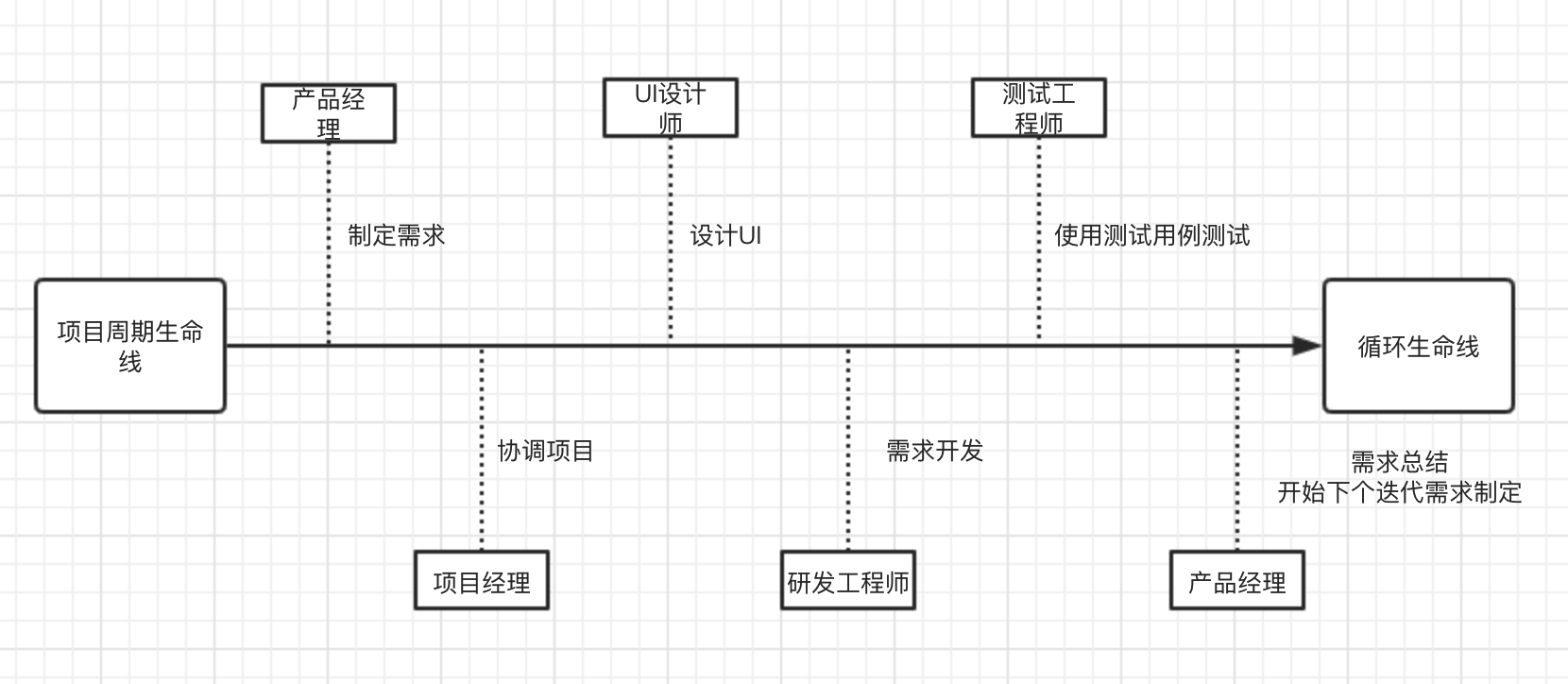
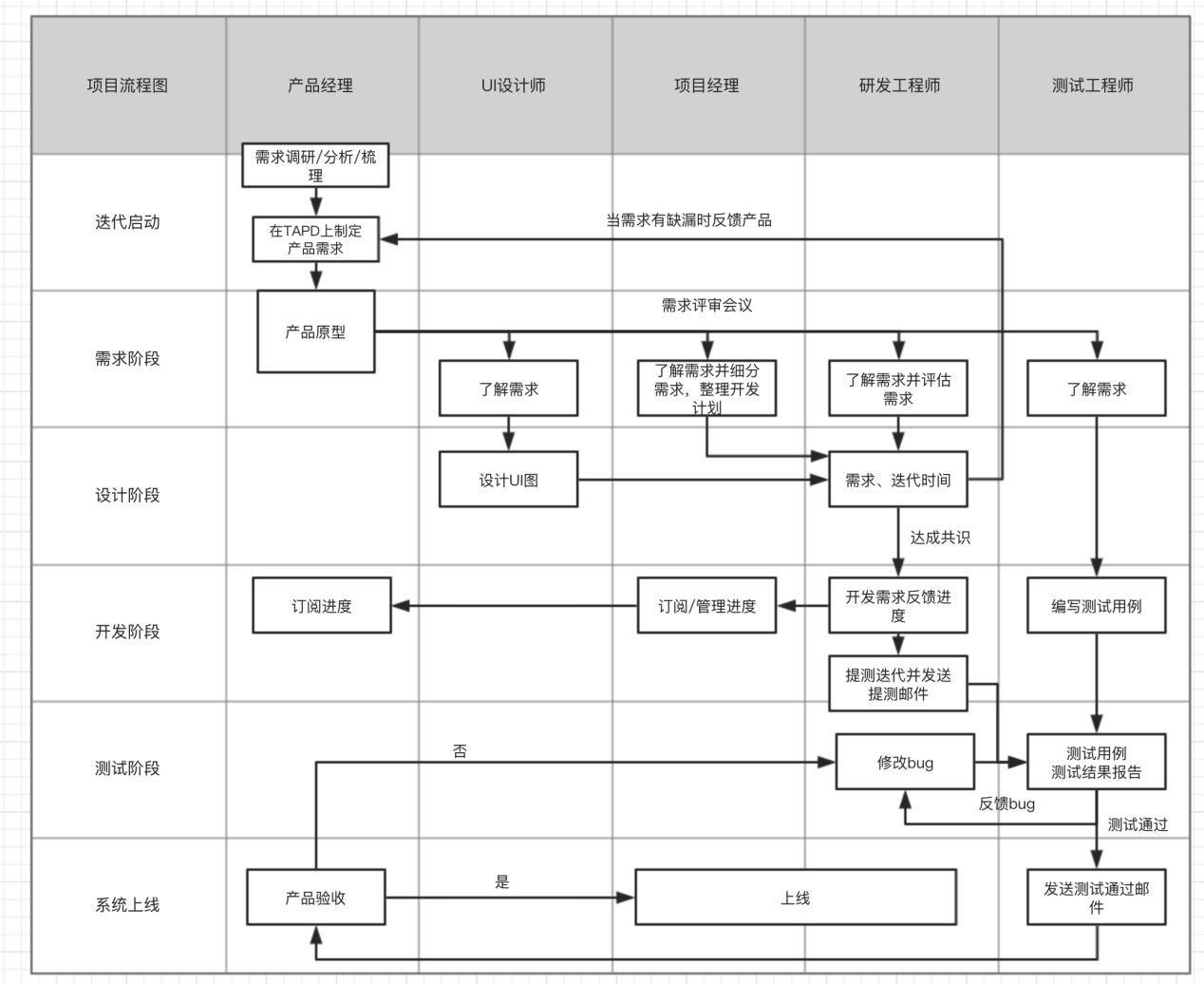
# 项目管理说明文档

**开发生命周期图**



**开发流程图**



**项目启动会**

项目启动会的目标是明确以下职责：

1、该产品开发项目的目标。目标不是孤立存在的，目标与计划相辅相成，目标指导计划，计划的有效性影响着目标的达成。所以在执行目标的时候，考虑清楚自己的行动计划，怎么做才能更有效地完成目标，是每个人都要详情清楚的问题，否则，目标越是不清晰或是过高，都会影响项目的实际结果。

2、说明项目目标

3、说明阶段划分

4、组织结构

5、管理流程关键事项

并将以上内容写入TAPD（最好是有固定格式和范文，让团队内部或者公司内部共同遵守规范），需要大家达成一致。对于关键角色任命，事前也需要听取相关领导和项目主要干系人的意见。

**用户需求**

软件开始开发前需要确定代价和所获得价值的对比，也就是 ROI（Return On investment），即是付出与回报，一旦确定需要创建，就需要安排一系列的资源来支撑这个软件的生存。该需求渠道目前主要设计为两种：

1. 是用户反馈通道的信息筛选整理需求。
2. 通过用户数据分析后整理需求。

为什么既要有用户需求，也要有产品需求？因为两者是有差异的，基本上用户需求主要由用户提出，对技术一般不描述，只描述产品目标。产品需求是根据用户需求转化而来的技术实现需求，需要针对用户提出的产品目标进行细分，总结出具体的每一个功能点，再针对每一个功能点细分为各种不同的操作流程，对每一个操作流程进行技术化定义。

用户需求和产品需求容易发生不一样，这是因为虽然大家都在谈需求，但是出发点可能不同，造成了双方关注点和思维方式不同。用户需求关注的是系统如何支持业务流程，背后的需求是“实现业务目标”。技术人员关注的是合理技术方案，背后的需求是“工作量”、“实现难度”和“系统性能”。

**产品需求**

我们需要弄清楚产品经理或项目需求提出者为什么要做这个项目？这是最本质的业务需求。需求分析确定的业务需求，都是从业务需求推导出来的，都必须为业务需求服务。

产品需求一般包括产品需求规格说明书和产品需求矩阵。产品需求矩阵一般按照子系统、功能集、执行单元的结构列出所有的功能需求，每列则对应每项功能的工作步骤以及每个步骤的工作量。

产品需求需要写在TAPD上按模块细分编写需求，需求如下：

1. 必须考虑周全且细化规则，并完善原型图，提供原型图路径。（举个例子，当产品思考一个完整需求需要花费一个小时编写完毕，开发需要两个小时完成开发，当产品只用了半个小时写出了未完善的需求文档时，开发用两个小时开发完后，产品发现还需要用剩下的半小时优化需求，则可能导致开发又使用多两个小时进行开发，大大增加了项目时间成本。）
2. 通知UI设计师在TAPD该需求下根据原型图设计补全蓝湖路径。
3. 进行相关技术人员需求评审。在需求评审会上，产品、技术详细评审需求是否完整，产品功能的正常场景是什么？是否形成闭环？异常场景是什么？是否考虑周全？

4、需求评审后，开发和测试负责人，分别编写技术方案和测试用例。技术方案评审，开发负责人拉上涉及到其他系统的负责人一起讨论，技术方案中必须要有业务流程图和时序图，业务流程图是为了梳理开发对业务的理解，是否和需求一致。时序图是了梳理本次需求涉及的系统交互。技术方案评审通过后，确认工作量和交付时间，反馈给产品，最后形成如下图。



**开发详细设计**

详细设计阶段就是依据概要设计阶段的分解，设计每个模块内的算法、流程，为每个模块完成的功能进行具体的描述，要把功能描述转变为精确的、结构化的过程描述。

1、各个模块可以分给不同的人去并行设计。设计者的工作对象是一个模块，根据概要设计赋予的局部任务和对外接口，设计并表达出模块的算法、流程、状态转换等内容。这里要注意，如果发现有结构调整（如分解出子模块等）的必要，必须反馈到项目经理处，评估工作耗时情况，而不能就地解决，不打招呼。详细设计文档最重要的部分是模块的流程图、状态图、局部变量及相应的文字说明等。2、为一个模块编写对应一篇详细设计文档。设计文档中可用流程图、思维导图、伪代码等。而详细设计的目的是描述某一个模块内部的处理流程、开发方法和编码技巧。一般来说，详细设计由项目简介、模块说明（具体说明每一个模块内部的流程、功能、逻辑、消耗以及未解决问题）、接口设计（包括内部接口和外部接口）、数据结构设计（包括物理结构和逻辑结构）、特殊处理等几个部分构成。软件的详细设计，最终是将软件系统的各个部分的具体设计方法、逻辑、功能采用文字方式进行表述。这样在实现过程中，编码人员原则上严格按此进行代码实现即可。

**编写代码**

编写代码可以遵循以下几点原则：

1. 优先做核心模块的压测，很多程序员，习惯把东西做完，然后等着快上线的时候才做性能测试，那么如果前面设计出了问题，这个就很头大了。
2. 后期上线时需要做性能测试。
3. 熟悉业务，需清楚业务压力集中点，业务请求重心集中点，遇到产品描述不清晰，需要自行提问。

4、确保过程可控：代码执行时一定要保持中间的输出，比如说，每处理 10 万条日志，写一条状态日志，记录处理的日志条目数和当前的执行时间。

5、目前代码质量较差但能完成业务的情况下，需要留下注释，并说明下一步优化的可能思路或可行方案。

6、尽量写出简单易懂的逻辑，如果逻辑真的很难在一个函数内完成，尝试切分。

7、使用熟悉、成熟的技术。使用新技术前，建议全面了解该技术的特征，适用范围，以及不适用的范围。

8、根据情况可编写单元测试、集成测试。

**代码审核**

在团队中进行代码审查（Code Review）可以提升代码质量，分享项目知识、明确责任，最终达到构建更好的软件、更好的团队。

代码审核及其重要，一般来说每周都要做一次代码审核。首先，代码审核有利于你跟踪项目进展情况，我们能真实地看到手下的人进展如何，并且更早发现他们是否误入歧途。有时候，TAPD上反馈的进度只是个人总结修改的进度，唯有直接看代码时才能掌控实际进度。在管理中，我认为代码审查是避免这种麻烦的最佳途径。

**系统化测试**

测试工程师需要对系统测试阶段做到以下几点：

1. 制定系统测试方案以及用例编写、功能性测试。
2. 对后台接口进行参数测试、性能测试、稳定性测试。
3. 验证需求分析确定的功能是否齐全并被正确实现，同时还要对安装、部署、兼容性、安全性、界面等非功能性需求进行测试。总结并反馈在TAPD上并关联处理负责人。
4. 环境测试流程依次以开发环境、测试环境、预生产环境、正式环境使用测试用例。

**产品发布**

1、产品发布是系统测试结束后的最后一步，需要系统测试工程师输出系统测试报告和产品经理输出批准产品发布（上线）报告后由项目经理统筹安排进行上线。

产品发布前需要通过产品发布说明会形式，对整个产品开发过程从立项开始回溯过程，指出整个过程中的不足点，总结经验，为下一个项目提供经验案例。这一会议可以通过正式会议形式召开，需要召集产品经理、主要开发人员、测试人员、上级领导等参与，准备充分，尽最大可能说清楚这个产品发布之后的效果、效益，为上线后的价值评估做准备。这一环节不可缺少，即便在互联网公司，迭代速度很快的情况下，这一环节也需要满足。

**过程复盘**

1、编写周报并捋清思虑和需求是一门自我提升的必修课。所有的总结，只有带着问题去思考才会有收获，这就是复盘。看清一个问题最好的方式，就是当处在一个问题的两个不同的角色中的自我思考。总结项目经验教训的目的，在于总结问题、分析原因，避免以后犯同样的错误，而不是追究谁的责任。

假设一个需求理解的缺陷，如果在需求阶段发现，修改一下可能只要一个小时，但是如果到了设计完成时发现这个缺陷，因为涉及的人员、文档增多，估计要一天时间，而如果等到代码都编写完成时才发现这个缺陷，可能需要十天八天了。如果缺陷没被发现，而是直接到了生产系统中呢？这就不是工作量的问题了，估计损失就难以估计了。在质量管理的理论中，缺陷每延迟一个阶段被发现，修复的代价就要乘上十倍。