持续探索是通过持续探索市场和客户需求，并为实现这些需求定义产品愿景，产品路线图和一系列特性，来促进创新及构建必要一致性的过程。

定义详解：产品愿景是整个项目的目标，好的产品远景能让项目团队了解产品的价值，建立共同的目标并激发团队士气。如何实现这些目标？好的产品愿景应该包含哪些方面？

* 产品是什么
* 产品将是什么
* 产品应该是什么
* 用户是谁
* 竞争对手是谁
* 如何满足用户需求并超越竞争对手

假设验证： 新的灵感始于假设——人们永远不会真正知道他们是否可行，一切假设均需验证。

* 精益创业思维：最小可行产品的定义有助于在投资过多之前迅速对假设进行评估。最小可行产品是能用来评估假设是否有效的最小单元
* 精益创业反馈环：把我的很多想法，排出一个次序来，然后开始去开发一个最小可行产品，然后投向市场，开始度量数据，然后分析数据结果，看到底能得到什么反馈，获得什么样的认知。

真正执行的时候，这个过程应该是逆向的：

* 首先，对于任何一个idea，我们都要把它想象成假设
* 其次，我们来思考该如何去验证，什么样的结果(认知)算是我们验证清楚了
* 然后我们再看该手机哪些数据，该怎么收集(度量)
* 有了MVP的想法后，最后一步才是把它搞出来(构建)

以造一部汽车为例，最开始的用户需求描述就是有轮子，能坐人，能控制方向。那按照第一行图片描述的过程，第一步我们先造几个轮子，然后用轮毂把轮子连接起来，为了能坐人，我们造个地盘，上面安上座椅，为了能控制方向，我们再造个方向盘，这样用户才算汽车的第一个版本完成了，这中间的跨度可能持续很久。如果按照下面的一行的流程，要造一部汽车，即要有轮子，又要能做人，又要能控制方向，那最小可实行的产品就是我造一个滑板车出来，上面按个车把，然后交付给用户。用户可能觉得不爽，你这做的也太简陋了，但是用户所说的功能勉强都有了，用户用了两天之后给你提意见，你这加速全靠脚，转弯还得停下来抬起车头，太麻烦了。然后根据用户反馈，我们把轮子变大，然后装上链条岸上脚踏板，用户可以直接踩脚踏板加速，车头用轮轴来和车轮相连，实现行进中转弯，然后再交付给用户。用户过两天又来提意见，上坡太累了，根据这个意见，我们装上发动机，变成摩托车，继续让用户体验，又过了一段时间，用户说摩托车长途坐着不舒服，而且刮风下雨啥的没个遮挡，我们再拿回来改进，变成四驱，也装上挡风玻璃，最终我们造出来用户满意的汽车，目标完成。

在造汽车的这个例子中，用户试用反馈就是数据，用户可以使用就是认知结果，滑板车，自行车，摩托车，汽车的不断迭代过程就是MVP的不断累积的结果。

协同研究：在执行过程中，需要不同的利益相关方进行合作，这就需要不断进行客户访谈，记录发现，不断参与贸易研究，以确定解决方案最实际的应用特征，为了拓宽思路，还要进行原始市场研究，分析和研究次级市场行业趋势，锁定新兴客户群，访谈行业分析师，并审查竞争解决方案。

架构设计：架构设计是人们对一个结构内的元素及元素间关系的一种主观映射的产物。架构设计是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计，这对于打造持续交付流水线至关重要。

* 云原生：容器化封装，动态管理，面向微服务
* 微服务：不再赘述
* 无服务器架构：通过多样的触发器与第三方云服务，客户端逻辑和调用云服务的能力集成起来，通常我们成为函数即服务
* 可发布型架构：解决方案的不同部分需要不同的发布策略。该方案必须设计为支持各种增量发布策略，并根据业务需求随着时间推移进行转换
* 测试型架构：不容易测试的系统也不会容易改变。以模块化方式设计和架构的系统支持持续的测试
* 分离部署和发布——为了持续部署，发布能力可能需要与部署到生产环境的工作分开。这种分离需要架构上的启动开关，这些启动开关将允许功能投入生产，而客户不可见。
* 运维架构——必须考虑运维需求。在每个应用程序和整个解决方案中建立遥测和日志功能。在高负载时或进行事故响应时，允许服务降级甚至移除。构建快速恢复和前向修复的能力。
* 威胁建模——信息安全方面的考虑应该尽早开始，确定威胁并据此提出建议的系统架构、基础架构和应用程序

愿景整合：将产品愿景，产品路线图，整合成带有优先级排序的特性待办列表。

* 电梯演讲——产品愿景是团队理解为何开发该特性的基础。
* 方案路线图——解决方案路线图它有助于对工作进行优先级排序。
* 效益优先级排序——特性必须经过优先级排序，开发才能有效。容量分配、投资范围和业务负责人持续参与的预算护栏对于优先排序至关重要，

总结：持续探索需求，意味着在需求相对清晰的情况下，即可开始迭代交付