DDD 是一个相对复杂的方法体系，它与传统的软件开发模式或者流程存在一定的差异。在实践 DDD 时，你可能会遇到一些困难。企业需要在研发模式上有一定的调整，同时项目团队也需要提升 DDD 的设计和技术能力，培养适合 DDD 成长的土壤。拔高一点看的话，我觉得你可能会遇到这样三个大坑，下面我来说一说我的看法。

1. **业务专家或领域专家的问题**

传统企业中业务人员是需求的主要提出者，但由于部门墙，他们很少会参与到软件设计和开发过程中。如果研发模式不调整，你不要奢望业务人员会主动加入到项目团队中，一起来完成领域建模。没有业务人员的参与，是不是就会觉得没有领域专家，不能领域建模了呢？其实并不是这样的。

对于成熟业务的领域建模，我们可以从团队需求人员或者经验丰富的设计或开发人员中，挑选出能够深刻理解业务内涵和业务管理要求的人员，担任领域专家完成领域建模。对于同时熟悉业务和面向对象设计的项目人员，这种设计经验尤其重要，他们可以利用面向对象的设计经验，更深刻地理解和识别出领域模型的领域对象和业务行为，有助于推进领域模型的设计。

而对于新的创业企业，他们面对的是从来没人做过的全新的业务和领域，没有任何可借鉴的经验，更不要提什么领域专家。对于这种情况，就需要团队一起经过更多次更细致的事件风暴，才能建立领域模型。当然建模过程离不开产品愿景分析，这个过程是确定和统一系统建设目标以及项目的核心竞争力在哪里。这种初创业务的领域模型往往需要经过多次迭代才能成型，不要奢望一次就可

以建立一个完美的领域模型。

2. 团队 DDD 的理念和技术能力问题

完成领域建模和微服务设计后，就要投入开发和测试了。这时你可能会发现一些开发人员，并不理解 DDD 设计方法，不知道什么是聚合、分层以及边界？也不知道服务的依赖以及层与层之间的职责边界是什么？

这样容易出现设计很精妙，而开发很糟糕的状况。遇到这种情况，除了要在项目团队普及 DDD 的知识和设计理念外，你还要让所有的项目成员尽早地参与到领域建模中，事件风暴的过程除了统一团队语言外，还可以让团队成员提前了解领域模型、设计要点和注意事项。

3. DDD 设计原则问题

DDD 基于各种考虑，有很多的设计原则，也用到了很多的设计模式。条条框框多了，很多人可能就会被束缚住，总是担心或犹豫这是不是原汁原味的 DDD。其实我们不必追求极致的 DDD，这样做反而会导致过度设计，增加开发复杂度和项目成本。

DDD 的设计原则或模式，是考虑了很多具体场景或者前提的。有的是为了解耦，如仓储服务、边界以及分层，有的则是为了保证数据一致性，如聚合根管理等。在理解了这些设计原则的根本原因后，有些场景你就可以灵活把握设计方法了，你可以突破一些原则，不必受限于条条框框，大胆选择最合适的方法。