DDD基本概念:https://www.cnblogs.com/netfocus/archive/2011/10/10/2204949.html

美团实践:https://tech.meituan.com/2017/12/22/ddd-in-practice.html

DDD不是架构，而是一种方法论（Methodology），因此不能把每个设计细节具象化。

DDD是一套体系，这就决定了它必须具有开放性，在这个体系中你可以用任何一种方法来解决这些问题。但如果这些关键问题如果没有具体落地的方法，可能会让团队无可适从。这其实也是DDD在许多项目中难以推行的部分原因。

Eric的著名书籍《领域驱动设计》的副标题是“软件核心复杂性的应对之道”，这个其实点出了DDD的来源和目标，很多因素会使软件的开发复杂化。软件是从现实世界到数字世界的一种建模和映射，软件复杂性的根本原因还是业务本身复杂性，软件开发者无法回避这种复杂性，所能做的是去控制这种复杂性。

领域驱动设计分为两个阶段：以一种领域专家、设计人员、开发人员都能理解的通用语言作为相互交流的工具，在交流的过程中发现领域概念，然后将这些概念设计成一个领域模型;第二个阶段是由领域模型驱动软件设计，用代码来实现该领域模型。  
 从DDD提出到开始流行，感觉经过了10年左右的时间，巧的是XP和敏捷从提出到流行也差不多10年左右的时间。可见一套方法论从出现到成熟确实是有一定规律并需要成长时间的。

按Martin Fowler在[PoEAA](https://www.amazon.cn/%E4%BC%81%E4%B8%9A%E5%BA%94%E7%94%A8%E6%9E%B6%E6%9E%84%E6%A8%A1%E5%BC%8F-%E7%A6%8F%E5%8B%92/dp/B003LBSRDM/ref=sr_1_1?s=books&ie=UTF8&qid=1460217008&sr=1-1&keywords=martin+fowler)一书中给了一个图。说明当软件在开发初期，以数据驱动的架构方式非常容易上手，但是随着业务的增长和项目的推进，软件开发和维护难度急剧升高。领域驱动设计则在项目初期就处在一个比较难以上手的位置，但是随着业务的增长和项目的推进，软件开发和维护难度平滑上升。

