## 「战术」访谈录：聊聊领域驱动设计

相信很多朋友对领域驱动设计会有这样或那样的困惑，比如领域驱动设计是什么？它在工作中有什么作用？为什么国内关于这方面的书籍少之又少？…… 为了解决这些疑惑，有幸邀请到专家张逸老师来聊聊领域驱动设计，下面是 GitChat 独家采访记录。

GitChat：在探讨领域驱动设计问题时，每个人都有每个人的认识，有的时候可能谁也无法说服对方，这时候该怎么办呢？

张逸：简单说，就是 **show me your code** 。不管领域设计做得怎么样，最终都是要落地的，看实现的效果最有说服力。当然，为了保证交流的顺畅与效率，代码这种形式可能容易让人迷失到纷繁复杂的细节中去，因此还有一种方式就是 **show me your model** 。

这里说的 model 就是领域模型。注意，团队在交流领域驱动设计问题时，不应该只是对建模活动的产出物进行讨论，建模的过程同样非常重要。现在诸如 Event Storming 等活动都非常强调利用可视化手段把业务专家与开发团队都包含进来，大家一块协作一块交流，并利用便利贴等工具直观地展现建模活动中的每一个步骤，可以更容易消除误会与分歧。

GitChat：可以谈谈领域驱动设计的流程吗？比如是先建模？还是做设计？以及应用的场景是什么？

张逸：领域驱动设计强调的是将分析、设计与实现统一到一个领域模型中来，同时又相对清晰地划分为战略设计和战术设计两个阶段。当然，这两个阶段并非瀑布式的，而是迭代和演进的过程。

我认同领域模型对分析、设计与实现的统一，这个思想没有问题。但在我亲身经历的项目中，我还是发现由于沟通角色与建模目标的不同，分析、设计与实现在三个不同的活动是无法完全统一的，就好像在重构时不能实现新功能，这三顶帽子自然也不能同时戴起来。因此，我在《领域驱动战术设计实践》课程中，清晰地将这三个活动称之为领域分析建模、领域设计建模与领域实现建模，它们各自的产出是领域分析模型、领域设计模型与领域实现模型，这三者合起来就是领域模型，而这个过程就是领域模型驱动设计。

之所以在模型驱动设计前面加上“领域”作为定语，是因为我认为二者不能划等号，例如采用数据模型的，同样是模型驱动设计。在《领域驱动战术设计实践》GitChat 课程中，我根据建模视角的不同，将其分别定义为数据模型驱动设计、服务模型驱动设计、领域模型驱动设计，并用了相当篇幅的内容分别介绍了这三种不同的模型驱动设计过程。

[点击了解《领域驱动设计实践（合订版）》](https://gitbook.cn/m/mazi/comp/column?columnId=5cdab7fb34b6ed1398fd8de7&utm_source=zyhd001)

GitChat：针对一些设计能力不足的开发团队，可以采用领域驱动设计来改进设计和编码质量吗？

张逸：我个人的观点，这二者之间有关系，但并非必要关系。领域驱动设计的关键不是设计能力，而是要抓住设计的驱动力，必须是领域，且必须要求领域专家参与到分析建模活动中来。

要说明的是，这个所谓“领域专家”不是一个头衔，也不是对技能级别的要求，它其实就是一个指代，代表“懂业务”的人：可以是客户，可以是 Product Owner，可以是业务分析师，可以是产品经理，也可以是懂业务的开发人员，甚至可以是一个负责业务分析的团队。

领域驱动设计能否成功，还是要看建模尤其是分析建模做得是否足够好，这其实是整个设计过程的上游。至于设计能力，则要看领域驱动设计与什么样的编程范式结合？常见的编程范式包括结构范式、对象范式和函数范式。因此这里的“设计能力不足”，究竟指的是哪方面的设计能力不足呢？

当然，从主流的领域驱动设计来看，主要采用的还是对象范式的设计思想与领域驱动设计结合，这就要求团队掌握基本的面向对象设计能力。这方面能力不足的团队，确实会影响到最终的设计和编码质量。这是必须要正视的问题， **因此我建议那些希望实践领域驱动设计的团队，不要忘了去提高团队的面向对象设计能力** 。

提升设计能力并非一朝一夕就可以做到。正是考虑到面向对象设计能力不足对领域驱动设计的影响，我在《领域驱动战术设计实践》GitChat 课程中尝试总结了一个 **相对固化** 的设计过程。这个过程结合了 DCI、职责驱动设计等设计方法，它不要求团队掌握太多面向对象设计思想、原则与模式，只要懂业务，完全可以以“知其然而不知其所以然”的方式去实践领域驱动设计。

这种方法不能让你的设计变得非常优秀，却可以保证你的设计不至于太糟糕，甚至可以说是不错的设计。

GitChat：应用服务与领域服务的区别是什么呢？

张逸：这个是老生常谈的问题了。从分层架构的角度看，应用服务属于应用层，领域服务属于领域层。应用层是一个包装的外观，按照该层的职责来说，应用服务根本就不该干业务的活儿，它只是一个对外公开的接口而已。

从业务粒度看，应用服务的每个公开方法会对应一个具有业务价值的业务场景或者说用例。领域服务则不然，它实现了业务功能，这个业务功能或者是无状态的，又或者是因为需要协调多个聚合，又或者需要和外部资源协作。

在针对业务场景驱动设计时，应用服务的一个方法往往会暴露给调用者，然后它再将该请求委派给领域层的对象。一般要求领域服务的粒度要小，这样可以避免设计为事务脚本的过程方式，也可以在一定程度上避免贫血模型。

总结：

* 应用服务：一组面向业务场景的业务外观方法，只是一个对外提供接口、对内分配职责的协作对象，属于应用层。
* 领域服务：一个领域服务对应最多一个业务场景，往往需要和聚合、Repository、甚至领域服务一起协作。

GitChat：已经上线的《领域驱动战略设计实践》与刚上线的《领域驱动战术设计实践》之间的区别是？有学习的先后顺序吗？

张逸：这两个课程刚好对应领域驱动设计的战略设计与战术设计。

前者强调系统层面的架构模式，包括限界上下文、上下文映射、分层架构等，可以运用这些模式对整个系统的领域进行“分而治之”，从而降低业务复杂度，同时围绕“领域”为核心，建立业务复杂度与技术复杂度的边界。

后者强调领域层面的设计模式，以“模型驱动设计”为主线，贯穿分析、设计与编码实现这三个不同的建模活动，并引入领域驱动设计的战术设计要素，如实体、值对象、领域服务、领域事件、聚合、资源库、工厂等。

当然在我的《领域驱动战术设计实践》课程中，我扩大了领域驱动战术设计的范畴，讲解了数据模型驱动与服务模型驱动，探讨了建模范式与编程范式之间的关系。同时，在设计过程中，我引入了职责驱动设计和 DCI 模式来阐释实体、值对象、领域服务与应用服务之间的协作关系。在编码实现过程中，我又引入了测试驱动开发来推进从设计模型到实现模型。

战略设计和战术设计并非单向的过程，而是一个迭代演进与不断融合的过程。整体来讲，前者更偏向于架构设计，后者更偏向于详细设计与编码。主要还是看读者的关注点与侧重点，并没有一个绝对的学习先后顺序。我个人还是建议先学习《领域驱动战略设计实践》，虽然它更偏向于理论，难度更高一些，但是它毕竟概括了领域驱动设计的全貌。

GitChat：「战略」这个课程分析了“EAS 系统”，在「战术」课程也介绍了“EAS 系统”，两者的侧重点有什么不同吗？

张逸：与战略设计和战术设计的侧重点一样，在「战术」课程中，会针对 EAS 系统进行领域建模，并最终对其进行编码实现。从内容来看，后者会更接地气一些，毕竟讲的是落地的实现。

将战略和战术的 EAS 系统案例结合起来，就是一个系统的完整设计案例了。

到这里，访谈录的内容就结束了，大家若对 DDD 的内容比较感兴趣的话，欢迎订阅本课程，有任何关于本课程内容的疑惑可在读者圈留言或加群交流~

[点击了解《领域驱动设计实践（合订版）》](https://gitbook.cn/m/mazi/comp/column?columnId=5cdab7fb34b6ed1398fd8de7&utm_source=zyhd001)

下一章

还没有评论



评论

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持