链滴

领域驱动设计 DDD 之限界上下文

作者：valarchie

原文链接：https://ld246.com/article/1574911806551

来源网站：链滴

<blockquote>
战术设计是从微观视角对单个微服务的编码设计，而战略设计是从宏观视角对多个微服务的交互计。从人体学来说，一个器官的内部构造属于战术设计，多个器官之间的协作属于战略设计。好比心器官之间的协作配合。
</blockquote>
<h4 id="为什么需要战略设计-">为什么需要战略设计？</h4>
假设我们在设计订单模型时，下单操作涉及到会员等相关规则，比如增加积分等操作。（此时我还未将订单子域和会员子域进行剥离）。其实一个用户的下单操作不需要与会员的积分产生强烈耦合一旦会员的积分制度发生变动，又得需要在订单子域中进行修改，而订单子域又是我们的核心域，频的对核心域进行更改的风险较大。那如果将会员体系从订单核心域剥离出去呢？我们便使得两个子域职能更为清晰并且解耦，这也体现了单一职责的设计原则。
假设一个开发小组既负责订单逻辑又要负责会员积分逻辑，则使得这个开发小组本身的职责变得杂。如果使将订单和会员拆分开来，通过上下文之间的交互（即微服务之间的交互）使得订单开发小成员、会员开发小组各自关心自己的领域（即各自只关心各自的业务逻辑）。
<blockquote>
好比流水线上的工人，有利于分工，大家专注于自己负责的工序。
</blockquote>
战略设计的元素主要有子域、限界上下文、上下文映射图。
<h4 id="领域-子域-限界上下文">领域、子域、限界上下文</h4>
从广义上讲，领域即是一个组织所做的事情以及其中所包含的一切。比如我们是一家网上生鲜超，我们这个公司的业务范围就是线上的生鲜营销。而它即是我们正面对的领域。那如果在这个领域中建一个统一大而全的模型，项目将会逐渐陷入混乱，更好的处理方式是在这一整个领域中再细分出子。好比我们生鲜超市公司里面分了很多部门例如财务部，采购部，营销部，库存部等部门，大家各司职，分工明确。如果公司不分门别类划分部门，只有一个部门的话，每个人都得会各个部门的一些工。人员工作就变得杂乱无章且复杂，而公司管理也逐渐失控。DDD 的思想是提出限界上下文的概念类比于不同的部门，采购部有采购部的模型，营销部有营销部的模型，模型只在限界上下文中变动，影响其他限界上下文，将变动的影响范围控制在单个限界上下文中。使得风险在可控范围。限界上下我们可以通俗的理解为微服务中的单个微服务，比如订单微服务，商品微服务等等。
试图去创建一个全功能的领域模型是非常困难的，并且最终很可能是失败的。我们可以试图去通战略设计，按实际功能将这些交织的模型划分成逻辑上相互分离的子域，从而在一定程度上减少系统复杂性。
<blockquote>
限界上下文不一定是我们自己开发的，比如可能某些公司会使用外部的云库存服务，那么这个库限界上下文就属于外部。
</blockquote>
<h4 id="子域与限界上下文的关系">子域与限界上下文的关系</h4>
<blockquote>
子域是否与限界上下文一一对应？
</blockquote>
不一定，比如我们的营销系统引入了会员优惠制度，但一开始我们的优惠规则非常简单，此时可将会员优惠作为限界上下文中的一个模块。等到这个会员模块的规则逻辑开始愈加复杂之后，我们再其进行剥离使其成为一个单独的限界上下文。
<h5 id="一般来说-一个子域对应一个限界上下文-">一般来说，一个子域对应一个限界上下文。</h5>
为什么需要这么多子域呢？因为一个概念在不同子域的关注点并不一致。
比如顾客的概念，在商品子域中，顾客的浏览记录，购买记录，偏好作为在商品子域的重要关注。而在订单子域中，顾客的会员等级，余额，优惠券并是该子域的重要关注点。将模型放置在一个特子域当中，才能使得该模型更为清晰。
<blockquote>
不同限界上下文如何进行互通有无？此时就需要上下文映射图。
</blockquote>
<h4 id="领域的种类划分">领域的种类划分</h4>
领域中的子域可分为核心域、支撑子域、通用子域。

核心域：公司主要的业务领域，比如生鲜的商品子域以及订单子域
支撑子域：公司的库存帮助公司完成销售。他们就属于支撑子域。
通用子域：会员子域，在许多的网上购物平台上都会使用到的会员体系。它属于通用子域。

<h5 id="核心域并非是绝对的">核心域并非是绝对的</h5>
比如在外卖软件中，商店的位置服务是属于支撑子域，外卖软件中的外卖订单域便是核心域。但店的位置服务是调用外部的地图服务商的 API，这种位置服务在该地图服务商的业务领域中则是核心。
<blockquote>
而不同的限界上下文为了解决某个问题而进行协作，如何协作呢？在上下文映射图中我们会继续到如何通过集成限界上下文的方式来完成不同通用语言间的映射。
</blockquote>
<h4 id="为什么需要限界上下文">为什么需要限界上下文</h4>
限界上下文是一个显式的边界，领域存在于这个边界之内。领域模型把通用语言表达成软件模型创建边界的原因在于，每一个模型概念，包括它的属性和操作，在边界之内都具有特殊的含义。
举个例子，比如用户模型在权限上下文中代表了某个用户的权限，而在会员上下文中代表了某个户的会员数据。那如果我们在整个领域中用一个用户模型既代表权限模型又代表会员模型的话，便会淆开发人员的关注点，大而全的模型是对单一职责设计的破坏，使得后期维护更加困难。所以将模型分到不同的限界上下文，使得模型与我们的通用语言更加贴切符合。
在我们中文当中也有类似的情况，我举个例子。包袱这个词大家肯定都听过。但是它是一个多义。在不同的语境下有不同的意思。例如：“请大家放心，不要有思想上的包袱。”这句话中包袱指的负担。而“相声中的包袱”则指的是笑料。不同的语境相当于不同的限界上下文，限界上下文起到对型的解释更加清晰的作用。
<blockquote>
单个限界上下文的内容包含我们之前讲到的关于战术设计的建模工具。
</blockquote>
有时，我们可以使用模块来避免创建一些微小的限界上下文。通过分析分散在不同限界上下文的务，你可能会发现，模块可以将多个限界上下文减少到一个。模块也可以用来拆分开发者的任务职责因此我们可以使用更加战术化的手段来避免过多的限界上下文。
<h5 id="关于DDD的理解各有不同-欢迎网友评论一起探讨-">关于 DDD 的理解各有不同，欢迎网评论一起探讨。</h5>