两步到位，快速找准Bounded Context

张逸 逸言 2015-03-12

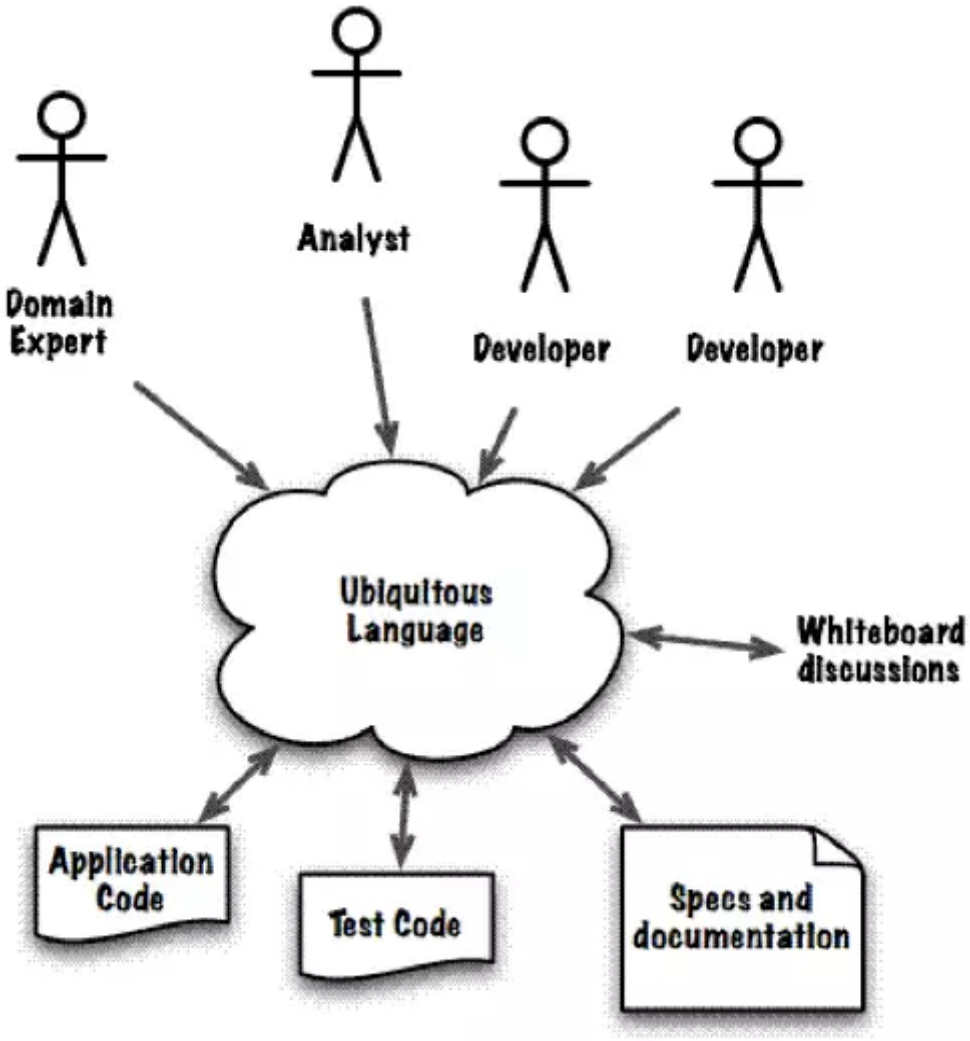
如何识别Bounded Context，在领域驱动设计方法学中无疑是一个挑战。我尝试利用可视化的用例图，通过两个步骤驱

在《初窥Bounded Context》一文中，我蜻蜓点水般讲述了Bounded Context是如何如何的重要，也尝试着堆积资料来阐述何谓Bounded Context。然后，我提出了问题，却没有回答。

我不是记者，不能只负责提出问题，而将答案抛给我看不见的对方。关于“如何确定或划分Bounded Context”，我确有诸多疑问与不解。用浪漫一点的话来讲，Bounded Context就像是爱情，懵懵懂懂之间你好像看到了她，如蝴蝶翩迁飞舞于花丛；若你心急火燎冲过去捕捉，非但会踏坏花丛，蝴蝶也被你莽撞的身影惊飞；高明的“捕猎者”，须得气定神闲从容不迫，预先设定好陷阱，让蝶儿自己扑扇进去。

我发现一些高明的设计者，正有这种“不求自来”的魅力与气度。记得我就这个问题咨询了Implementing Domain-Driven Design的作者Vaughn Vernon，他的回答很酷：“By Experience”。这或许就是所谓高手的气度吧。然而，对于一种方法学（Methodology）而言，这样的回答是不负责任的。若缺乏具体而有效的指导，就好像预测未来，看到的是一片茫然。

Bounded Context与领域相关。在识别Bounded Context之前，我们必须要掌握与领域有关的知识。DDD中提出统一语言（ubiquitous language），但它的主要目的是为团队的不同角色达成领域语言的共识，从而消除分歧或误解，并不能必然成为Bounded Context的输入。



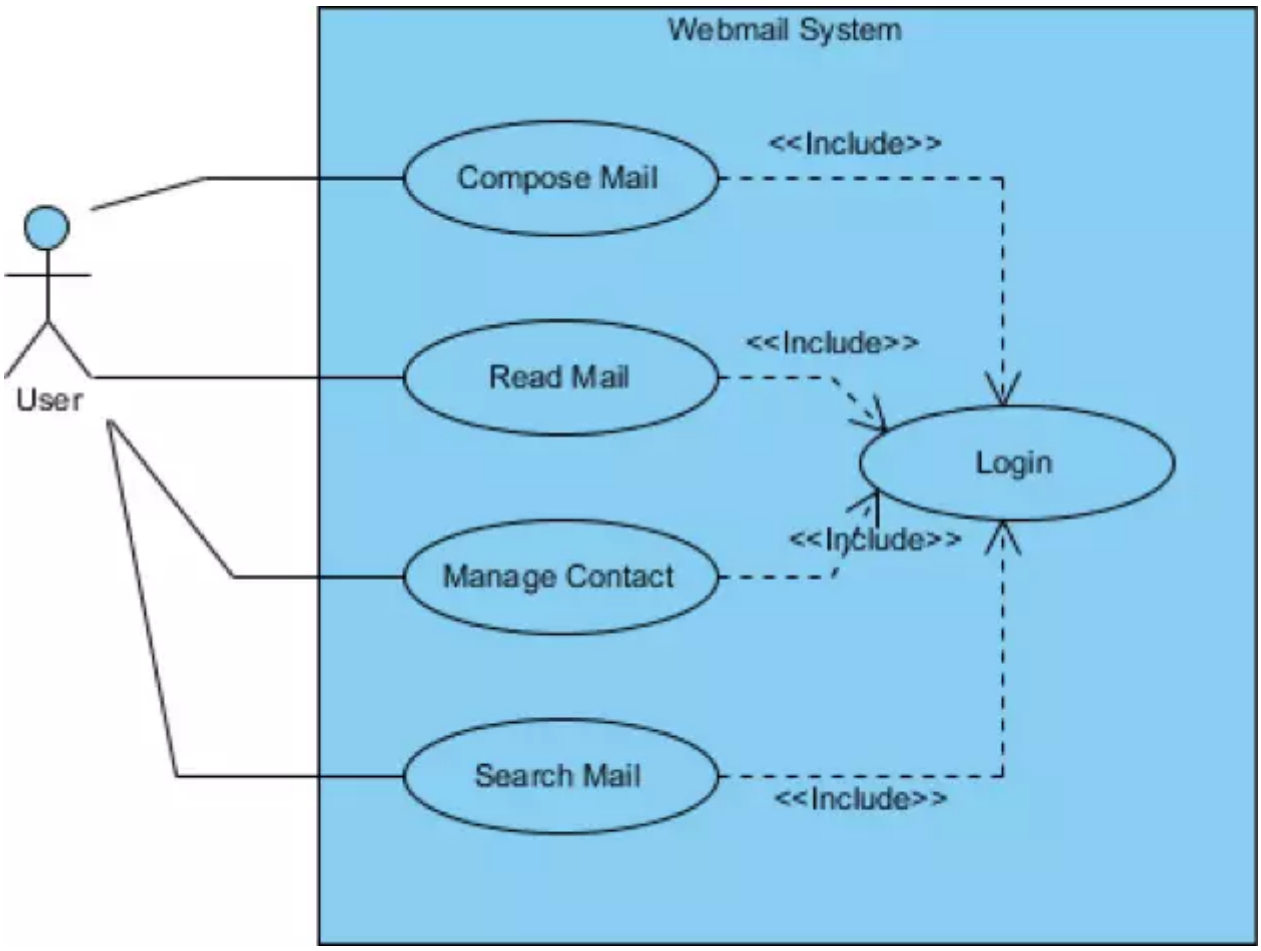
△ 统一语言在于为不同角色达成一致

要理解“领域”，一种方式是通过“场景”。相较于领域，“场景”这个词更像是从抽象的三维球体中，切割出来具体可见的一片。以这一片场景为舞台，可上演角色之间的悲欢离合。每个角色的行为皆在业务流程的指引下，同时受到业务规则的约束。当我们在描述场景时，就好像在讲故事，又好似在拍电影。

场景与“6W”模型有关，即组成一片场景的要素包括Who，What，Why，Where，When与hoW。从战略设计的角度看场景，我们并不需要知道该场景究竟该如何（hoW）实现，以及场景内部角色之间的具体交互流程（When），这是战术设计需要关心的。那么，什么才是适合我们去了解领域中Who，What，Why，Where的模型呢？

我从传统的用例（Use Case）中获得了灵感，严格说应该是用例图（Use Case Diagram），因为在战略设计阶段，我们希望能快速灵便地获得架构草图，然后开始短小轻快的迭代。冗长的用例描述，已经显得有些不合时宜了。

为何用例图符合我们的期待呢？组成一个用例图的要素包含参与者（Actor，代表前面提及的Who），用例（Use Case，代表前面提及的What），用例关系（使用、包含、扩展、泛化、特化）以及边界（Boundary，代表前面提及的Where）。如下图所示：



△ 用例图

至于Why，是要通过我们不断地询问为什么来驱动出Use Case对于Actor存在的价值。在用例图中看似没有体现出所谓“价值（Value）”，但是，当我们在思考用例关系时，正是价值悄悄地在影响着我们的设计。思考上图的Login用例，为何没有与User之间产生用例关系，难道说用户不需要登录吗？它与Compose Mail、Read Mail之类的用例有何区别？

若从价值着手，答案就浮出水面了。因为对于用户而言，登录这个行为（或功能）是没有任何价值的。即使需要登录，也是为了编写邮件、搜索邮件。当我们在设计用例图时，必须思考这种价值是否存在，否则用例图就可能存在谬误。

一个设计合理的Bounded Context，必须是高内聚、松耦合的。划分内聚与耦合的边界，就意味着我们需要去判别BC之间领域对象之间依赖的强弱关系。这种边界与用例的边界是暗合的。在上图中，我们看到用例的边界为Webmail System，这意味着边界内的用例都是与Webmail相关的功能。如果我们要设计的系统除了与Webmail有关，还牵涉到大量的安全认证、账户管理等功能，则从用例边界的角度来讲，Login用例就不应该划归到Webmail System边界中。

用例图仅仅是手段，而非目的。这就引申出另一个问题：该怎么运用用例图来识别Bounded Context？

我们首先要认识到任何手段都无法帮助我们一蹴而就地获得答案，软件设计尤其如此，需要不断地演化、迭代，才能做到恰如其分，也只是做到恰如其分而已。不要奢望完美！

我将用例图的手段分为两个步骤：

- 步骤一：从参与者出发，识别主要用例，包括主要的包含用例和扩展用例；
- 步骤二：根据语义相关性与功能相关性对用例进行分类，从而识别出边界，并为边界命名。

步骤一能够帮助我们识别用例。我的理念是希望通过堆砌出来的用例，向上驱动出最终的Bounded Context，而非借助经验捕风捉影地凭空想象。沉落到相对低层次的用例，我们就有了相对具体的驱动方法：通过参与者的驱动。如此再往前推，就变成对系统参与者的寻找。

我们要正确理解参与者的概念。它指的是角色，而非具体的人或其他事物。这个角色又是由用例来决定的。例如用例为“Purchase Product”，则对应的参与者为Customer或者Buyer；若用例为“Comment Product”，则参与者就不再是Buyer或Seller，而应该是Commentator，因为在对产品进行评论时，扮演的角色不再具有购买或销售的语义。

因而，不同的参与者观察系统的角度亦是不同的。正所谓“横看成岭侧成峰，远近高低各不同”，你所处的位置，决定你看到的风景；你扮演的角色，决定了参与其中的业务行为。

我之所以决定采用用例图的形式，还在于它体现了“可视化”的价值。绘制用例图时，一定要干净利落，保证用例图的简洁与清晰。我曾经看过那种如蜘蛛网似的复杂用例图，如风中的一片凌乱，阅读这样的用例图，只会陷入其中，哪还有什么参考价值？

要保证用例图的简洁与清晰，一方面需要确定用例的层次，就像建筑设计需要事先谋划布局一般，我们要学会化简，把那些过于细节的用例剔除掉。另一方面则需要确保用例的专注度。就步骤一而言，应该为每个参与者绘制单独的用例图，从而避免出现过多交叉的用例。

如果多个参与者使用了相同的用例，应该怎么办？为了能够按照参与者拆分用例，可以考虑引入重复，即绘制重复的用例，然后将其分到不同参与者所属的用例图中。例如在一个培训系统中，参与者Owner与Assignee都使用了“Cancel Course”的用例，但是在步骤一中，我们可以绘制两个“Cancel Course”用例，分别放到属于Owner和Assignee的用例图中。

步骤二的核心思想是分类，从而水到渠成地引出“边界”。分类的基础是用例已经存在，而前提还在于我们对用例有合适的命名。用例的命名必须符合领域的语义，最好与领域专家合作，或者脱胎于统一语言。用例名称应该是动宾短语，因为它是一种功能行为，且需得有操作的目标。

语义相关性主要来自于用例描述的宾语。例如从直觉上，电子商务系统中的Search Product与Purchase Product，View Product Detail就应该属于同一个分类，因为它们的语义都与Product有关。

但是，语义相关并不必然确定它们属于同一个分类，反之亦然。此时，还需要考虑用例之间的功能相关性。无论是语义相关，还是功能相关，目的都是保障边界内的用例是高内聚的。所以在电子商务系统中，Place Order与Payment Online虽然功能相关，但如果希望电子商务系统不要与支付产生强耦合关系，就需要将这两个用例分到不同的边界中。

步骤二是对步骤一结果的一次重构。因此，在分类时我们还需要判断之前识别的用例是否合理，同时还要识别一些可能遗漏的用例。

为边界命名也是一种设计的驱动力。当你发现无法找到准确的名字来概括该边界内的所有用例，又或者命名变得很长，需要and等连词来连接时，则说明边界内的用例可能不够内聚，换言之，它违背了“单一职责原则”。

一旦识别并命名了边界，即可映射为DDD中的Bounded Context。接下来，就可以抛开用例的概念，进入领域驱动设计的世界了。
