【系统架构】可视化与领域驱动设计

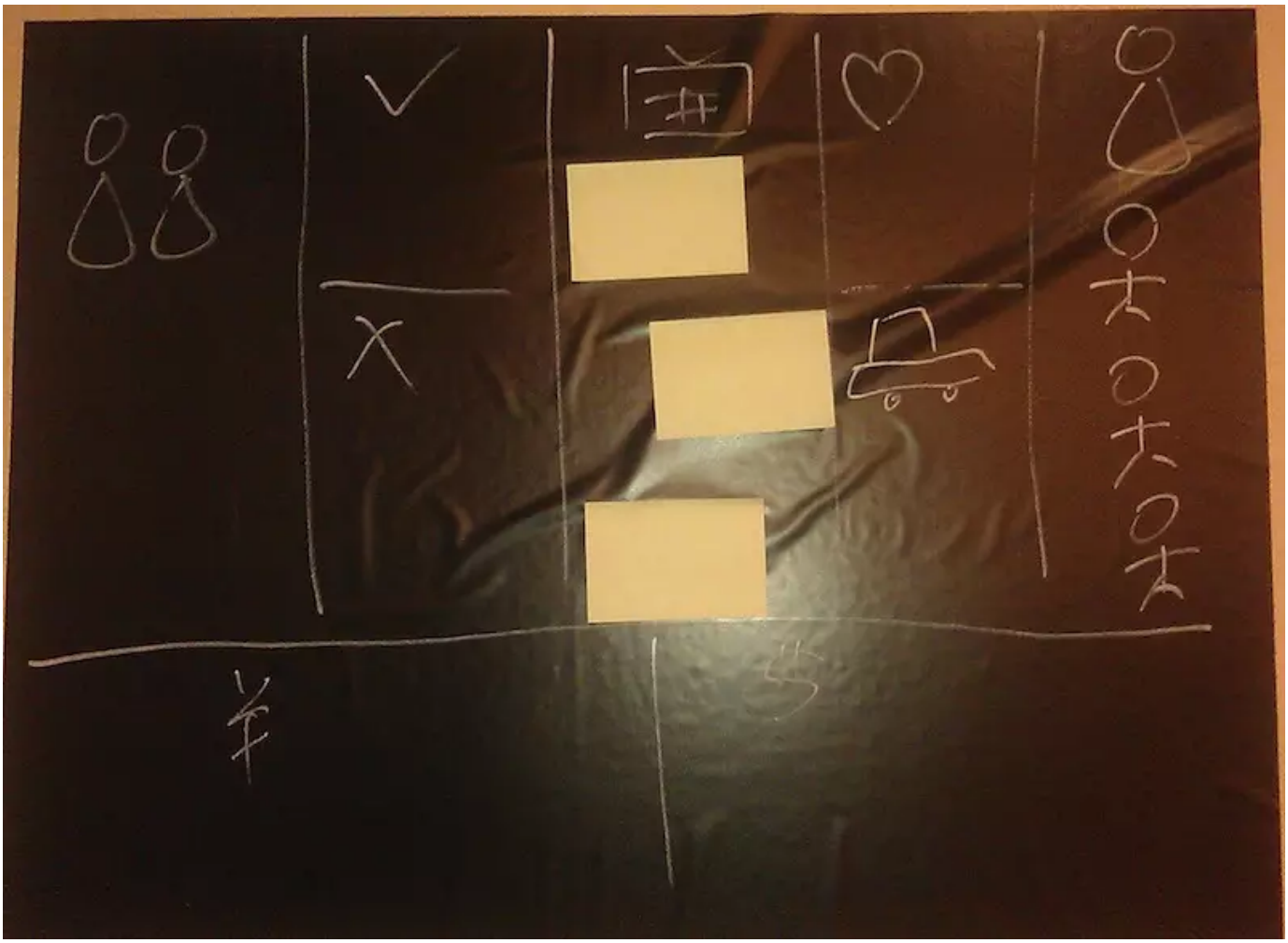
张逸 逸言 2014-08-15

从DDD的角度，领域逻辑的分析可以运用战略方法Bounded Context。可是，一个问题是：如何获得Bounded Context ？

我查看了许多关于Bounded Context的书籍与文章，虽然都着重强调了它的重要性，也给出了一些实例，却对如何从需求——>Boundex Context这一点上语焉不详。

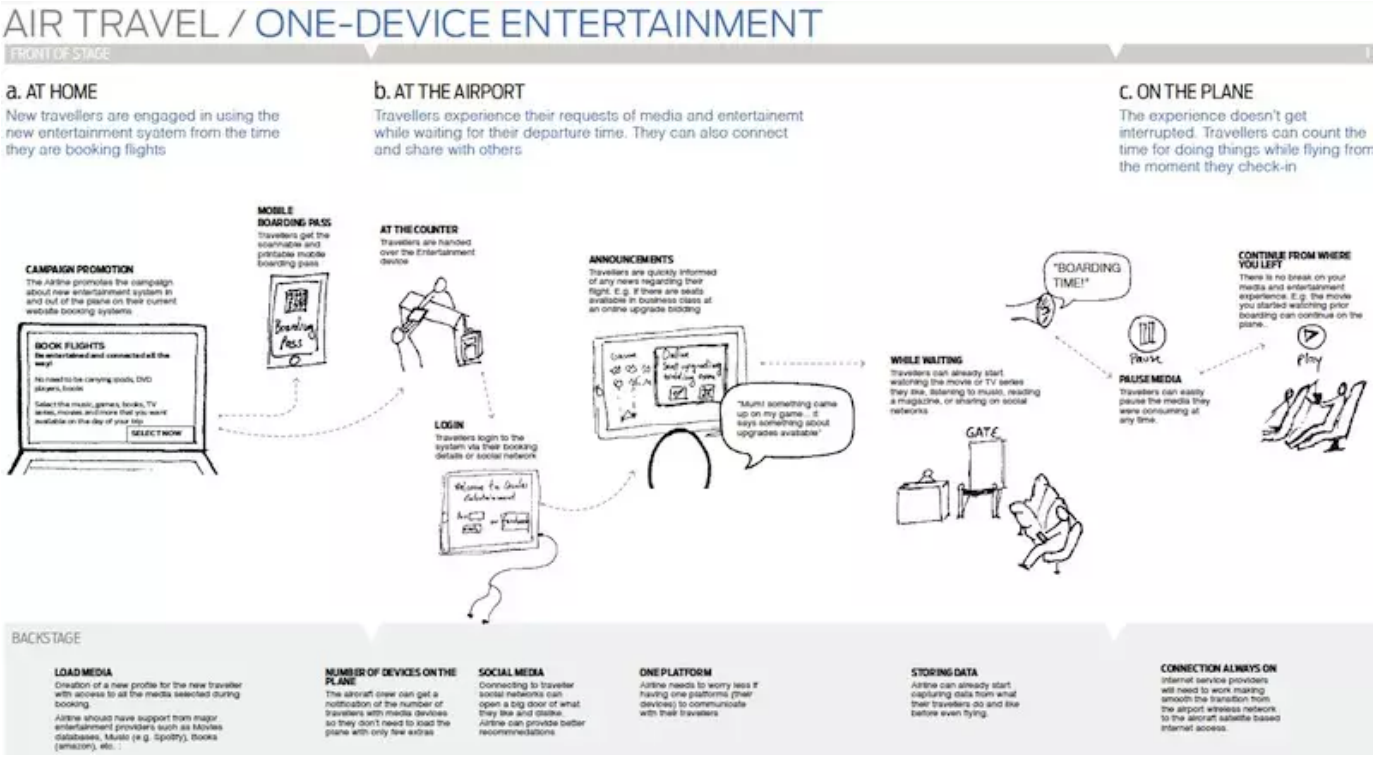
一个初步设想

我的初步设想是通过绘制场景图（但并不成熟）。我认为有三种绘制场景图的方式：商业画布，体验地图和流程图。我认为，商业画布可以作为需求分析（尤其针对初创产品）的起点。商业画布如下图所示：



采用这种规范化的方式来推导商业模型，可以激发我们的灵感，理清我们的思路，以便我们思考为何要做这个产品，产品应该具备哪些功能。结合优点和缺点、成本等因素，我们可以藉此判断和决策功能的优先级，从而得到MVP。这个过程需要大量运用即时贴，让

整个商业模型呈现。经过取舍后，就可以针对产品绘制场景图。此时，场景图可以采用Experience Map或流程图来体现。Experience Map的例子如下图所示：

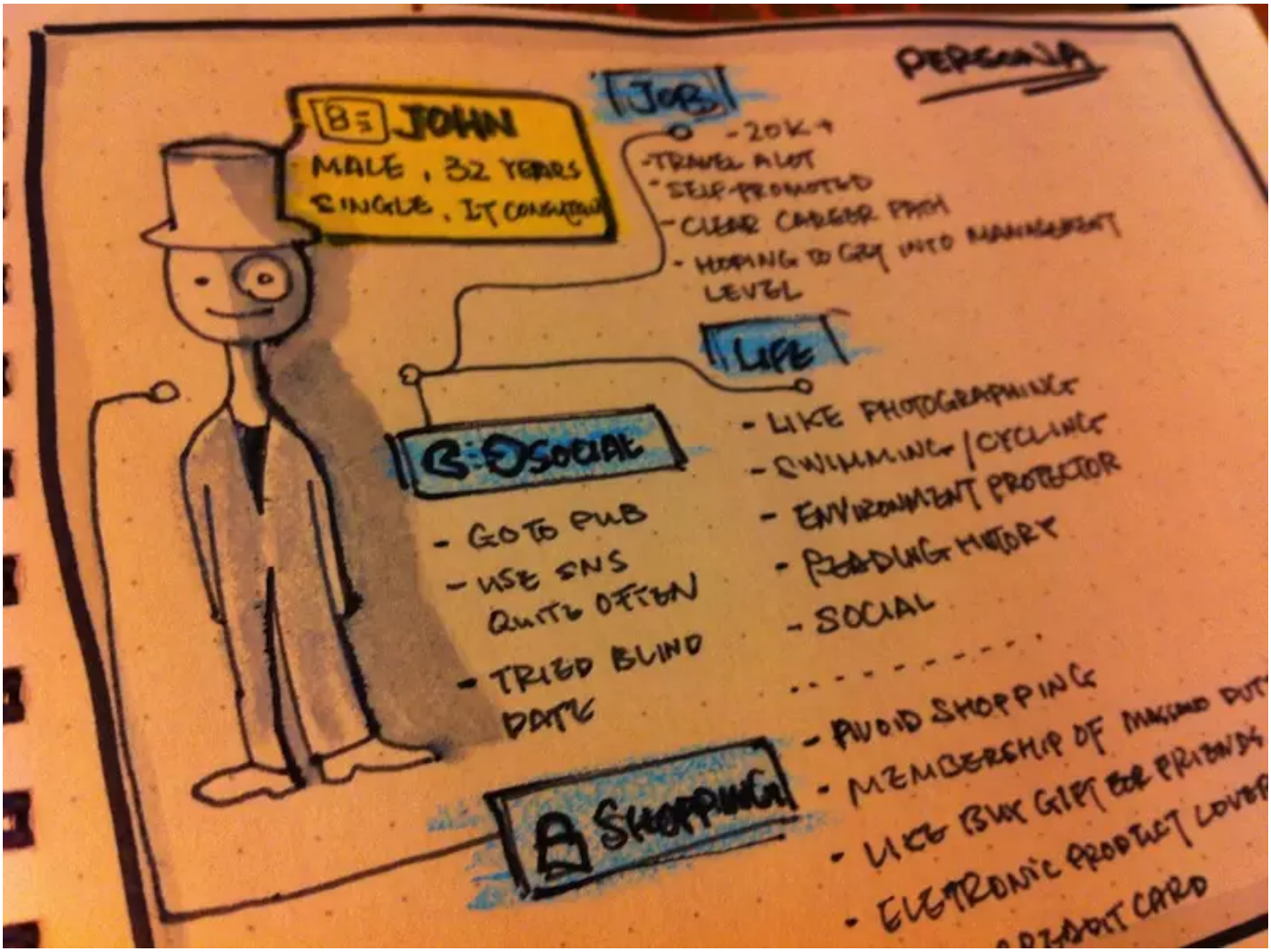


由于商业画布本身提供了“客户”项，我们应该创建Persona，找准人物角色的特征来“搜寻”需求。绘制了场景图后，就能够确定用例了，此时，可辅以ATDD帮助确定Story。在确定了用例后，可以识别Bounded Context，并通过Context Map确定上下文之间的关系。

就我个人感觉，体验地图还是从Persona的角度设想系统如何使用，考虑它的用户体验。它其实符合“场景”的概念。这里可能还是要考虑：在一个完整的场景中，需要哪些参与者？但是，即使从粗粒度的角度出发，场景都可能存在多个，可能需要绘制多个场景图来逐步提炼Bounded Context。

关于如何运用Persona，我的同事熊子川在他的博客《XD关键字5：Persona》中已有详细介绍，同样在他的博客《Agile UX内容策略工作坊》中提出的“消费者建模”实践，指出：

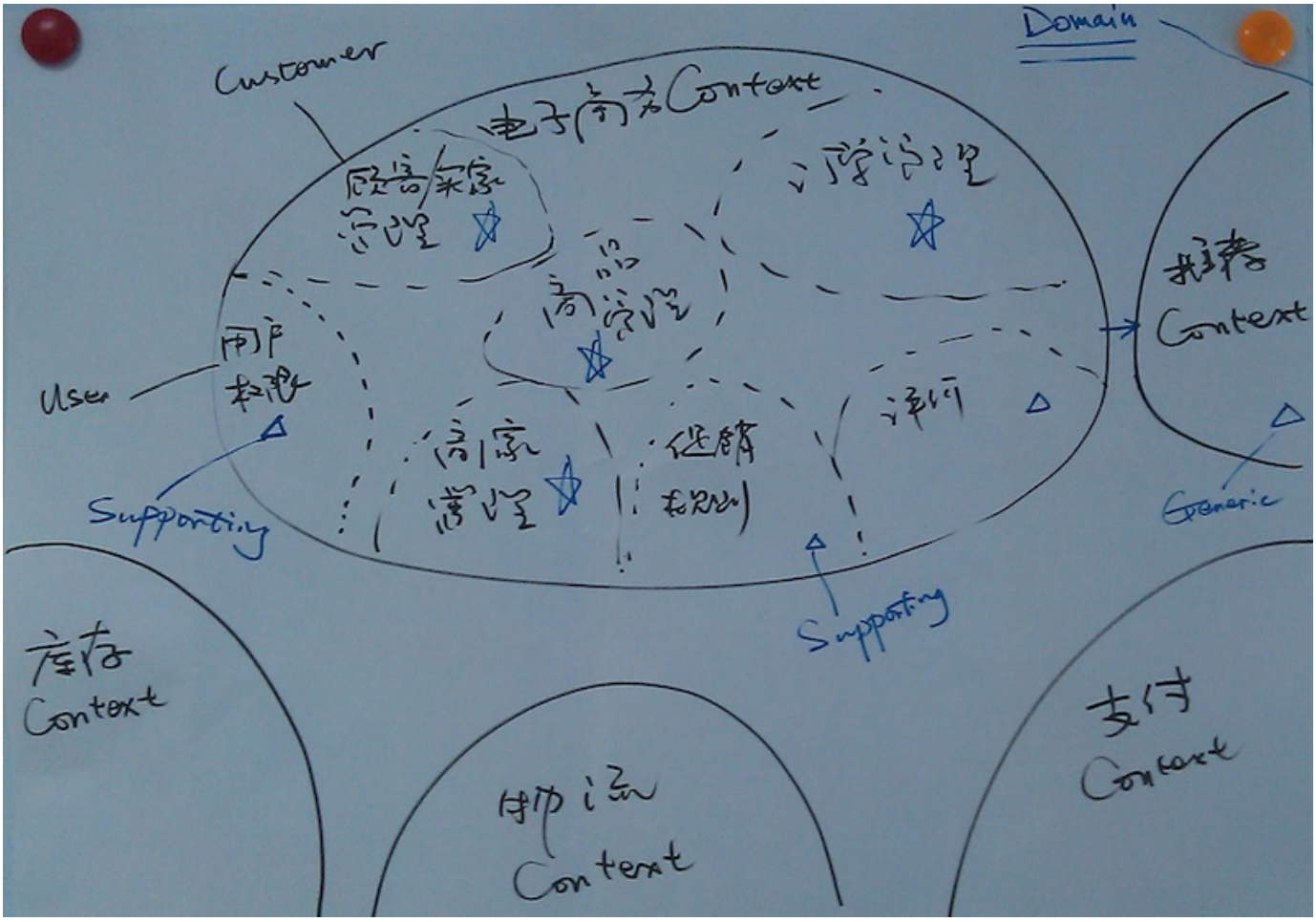
> 为了更好的理解我们选择的目标消费者，我们需要对消费者进行完整的建模，即Persona。越接近于真实的Persona帮助我们更好的理解其用户目标……Persona的重要产出物是一系列用户目标，对于同一个Persona，用户目标可能有不同，有些目标是基础核心目标，有些则是衍生性的，例如一个访问网站潜在投资者的核心目标可能是了解成为投资者的过程，而衍生性目标可能是获得一些关于公司历史信息增加信任度。



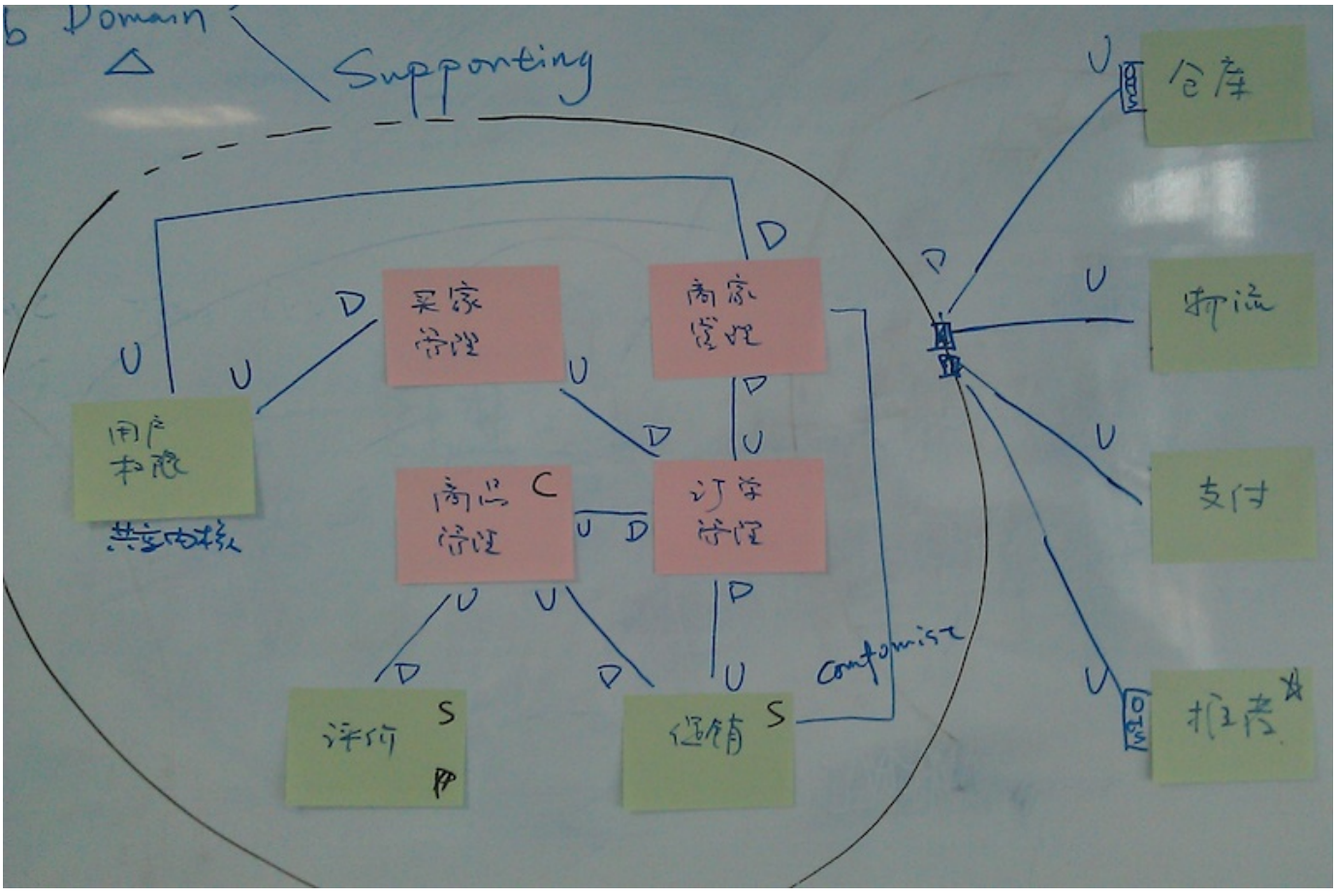
说明：本图摘自熊子川博客

获得Context并划分领域

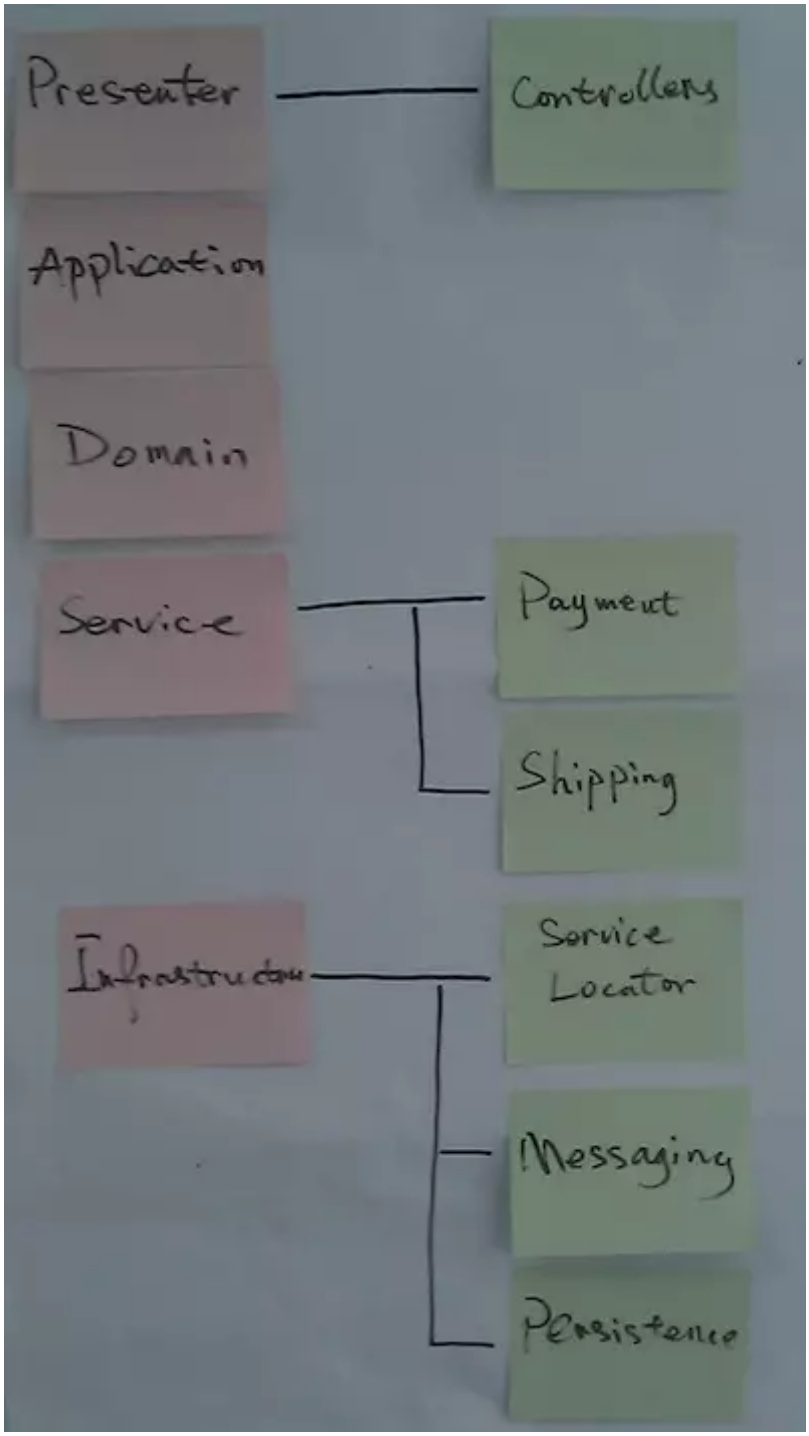
假设我们要开发一个电子商务网站，我们就可以通过商业画布来驱动出这个产品应该具有哪些功能，它的客户有哪些等，在绘制了场景图后，可以初步得到这样的Bounded Context:



然后，我利用Context Map得到了各个上下文之间的关系：

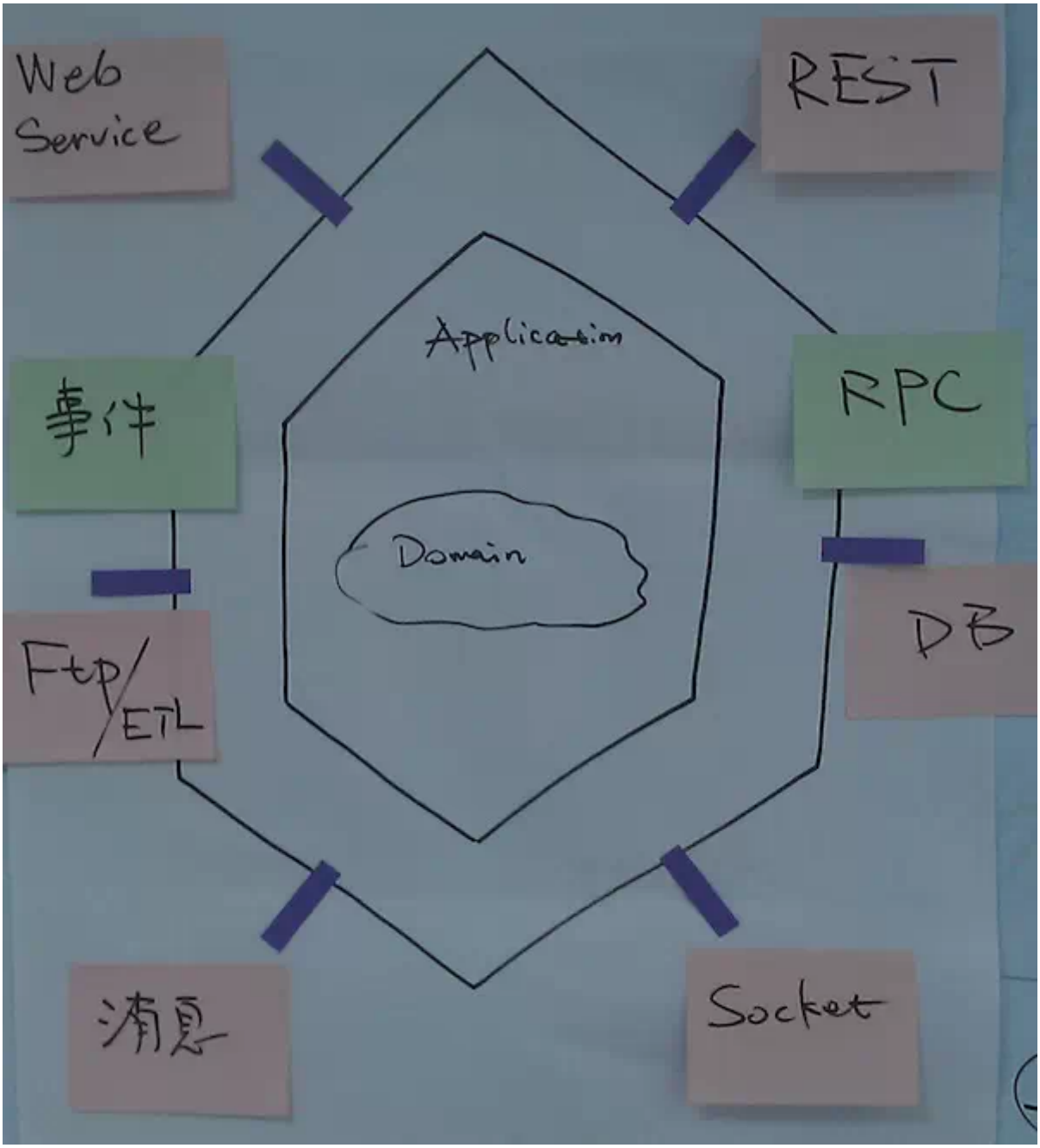


这样，一个包图的获得就水到渠成了：

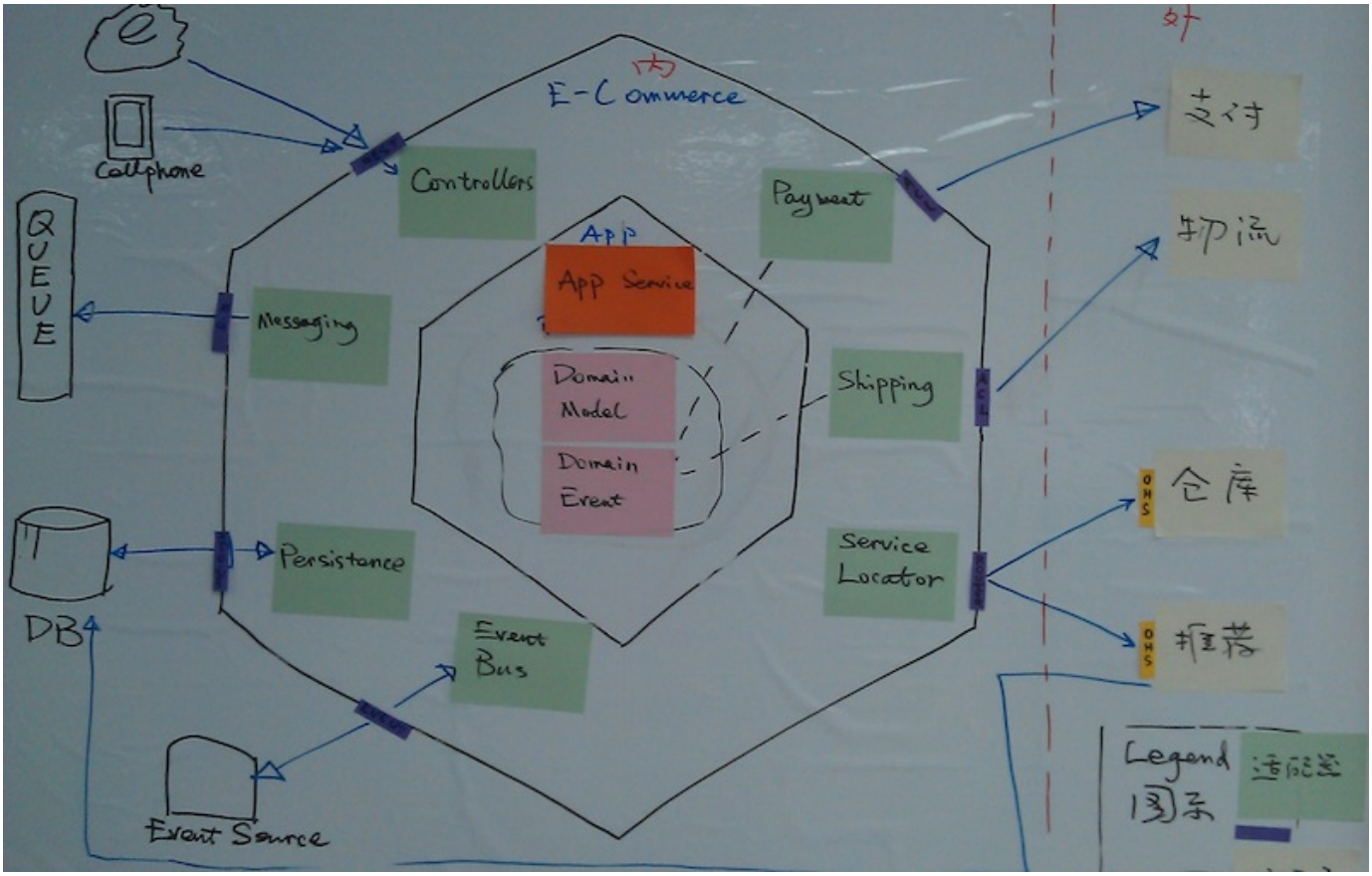


六边形架构

在识别了Bounded Context以及Context之间的关系后，我们可以运用Hexagon架构（Cockburn提出的六边形架构）来展现系统的整体架构。Hexagon架构并不深入关注内部边界中领域部分，仅仅是简单的划分为Application与Domain两层。但它有助于我们获得基础设施层以及相关集成点的包结构。我们要合理地运用六边形架构。它更贴近应用逻辑架构，并可以驱动我们去发现诸多集成点，寻找集成模式。内外边界的分离也有助于我们将业务逻辑与应用逻辑分离开。这实际上符合"关注点分离"的架构原则。下图展现了六边形架构中常见的Port与Adapter：



我对“可视化架构”的理解，还是要希望多通过即时贴、白板等工具来实现可视化，而非通过绘图。至少，绘图不应该成为主要的驱动力，否则，开发人员很难接受。例如，下图就是我运用Hexagon架构，并结合可视化手段分析该电子商务系统得到的应用逻辑架构，它很好地一个展现了Hexagon架构的可视化手法。



在这个图中，直观地展现了如何与外部的支付系统以及物流系统的集成。例如，图中展现的Port实际上为防腐层（ACL）。为何要建立这样的一个防腐层呢，原因在于：支付与物流常常存在多个供应商，因而需要解除对供应商的绑定，并避免供应商系统的变化造成对电子商务系统的腐蚀。这是切合实际的决策。

一个实例：仓库管理流程控制系统

这个电子商务系统需要与仓库管理系统集成。恰好在《面向模式的软件架构》卷四的第35页，给出了一个仓库管理流程控制系统的案例。书中描述的非功能性需求，即所谓质量属性包括：

- 分布性。仓库管理流程控制系统天生就是分布式的。
- 性能。仓库管理流程控制系统不是一个“绝对的”实时系统，但性能仍与业务息息相关。对系统有整体的吞吐量要求，因此系统必须确保所有的运输指令能够被及时而有效地运行。
- 可伸缩性。不同仓库其大小可能会有很大的不同，因此仓库管理流程控制系统必须能既支持只有几千个箱子的小仓库，又要支持超过一百万个箱子的大仓库。
- 可用性。许多仓库操作采用三班倒的24/7模式工作，因此可用性是仓库管理流程控制系统对业务案例支持的关键因素。

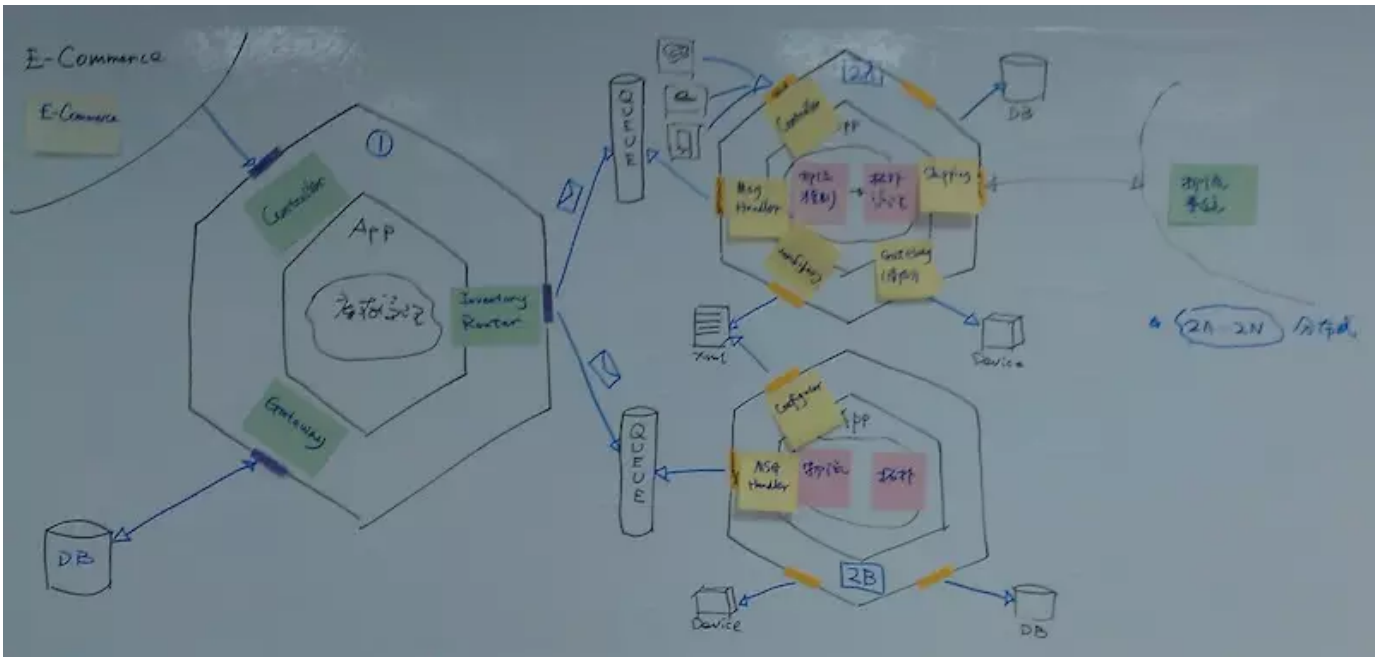
假设要设计这样的系统以支持这些质量属性。对于分布式而言，书中提出的解决方案是传统的分布式系统解决方案，即引入Broker模式，在本地建立对远程对象的代理。而对于

支持并发的领域对象访问而言，则采用了Active Object模式，并引入Leader/Followers并发模型来获得可扩展。

我没有打算引入这么复杂的模式，而仅仅是通过引入消息队列，并为消息队列引入路由的方式，来实现系统的分布式。这其中当然会用到经典的Publisher-Subscriber模式。我对领域逻辑进行了识别，将整个仓库管理流程控制系统的领域逻辑分为三个Bounded Context：

- 库存管理
- 物流控制
- 拓扑管理

整个架构如下图所示：



对于库存管理而言，我认为它主要支持商品存放信息的数据管理，即获得商品数量、存放位置以及更新这些信息。对于该上下文而言，操作本身比较简单，且耗时较短。若出现大规模并发，其瓶颈也不在于获取或更新仓库信息（当然需要通过测试数据验证），而在于客户下订单后向仓库管理流程控制系统发起的发货请求。

我将发货请求放到了物流控制上下文中，除此之外，它还包括收货以及订单管理等。同时，对于物流控制与拓扑管理功能，基本上与具体的仓库形成了一一对应关系。此外，对于发货请求（或收货请求），并不要求很强的实时性，这使得对这些请求的异步处理成为可能。

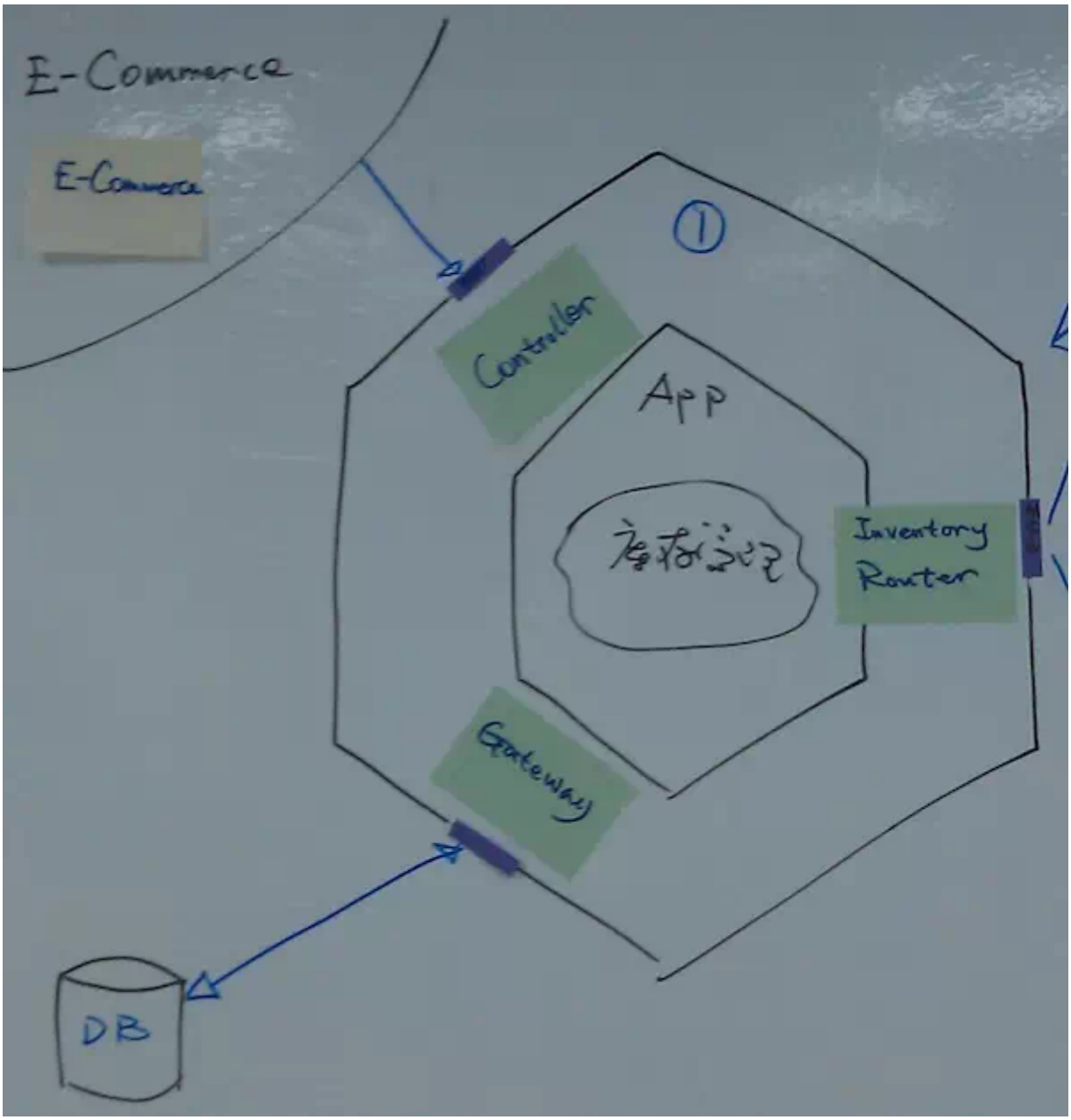
物流控制由于牵涉到收货和运货，需要控制仓库的相关设备，并按照仓库的拓扑结构设定设备的路由。这说明物流控制与拓扑控制存在上下游关系，拓扑控制是上游。这两个上下文可以是Customer-Provider的关系。但它们之间不应该存在物理边界。因此，我将这

两个上下文放到了同一个六边形中，而将库存管理放到了另一个单独的六边形中，以便于它们各自独立的可伸缩。

在库存管理与物流控制六边形之间，我引入消息队列来应对从库存管理子系统中转发而来的发货请求（发货请求实则又来自于E-Commerce的订单请求）。原则上，我针对一个物理的仓库建立一个单独的消息队列，因此库存管理在发送发货请求时，会根据商品的存放位置以及用户请求的IP地址，获得最优的仓库信息，然后通过Router将消息转发到正确的消息队列中。

一旦收到消息，物流控制系统作为消息队列的订阅者（或侦听器）就可以即使处理信息，进行后续的处理。

针对库存管理而言，我认为它是一个独立的物理边界，因此在可视化手段中，我展现为一个单独的库存管理六边形，如下图所示：

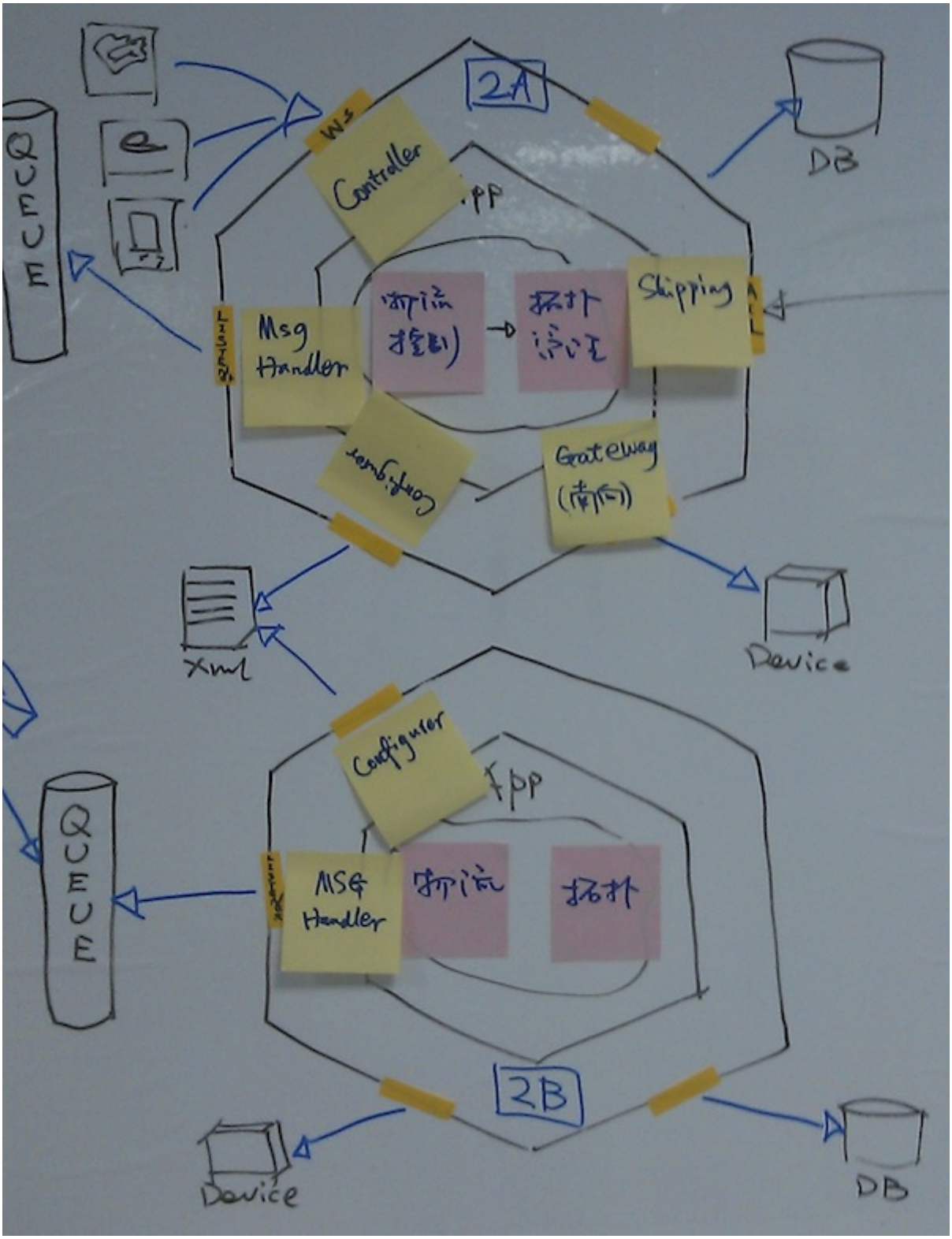


建立了针对REST服务的端口，对应的适配器为Controller，其目的是支持E-Commerce系统。事实上，我们对E-Commerce系统进行过分析，获得的六边形架构正好与此对接。

建立了针对DB的端口，对应的适配器为DB Gateway，它负责访问库存管理自身的数据库。数据库持久化的消息包括商品的基本信息如SKU、商品名、数量等，以及商品存放的仓库名。

建立了针对Queue的端口，对应的适配器为Message Router，负责将发货请求消息路由到正确的消息队列。

物流控制与拓扑管理放在同一个边界中，它是高度可伸缩的独立系统，为展现它的可伸缩性以及它与库存管理之间的集成，我在可视化手段中，展现出两个独立的六边形，如下图所示：



针对Queue的侦听器端口，对应的适配器为Message Handler。若有必要，如为了更好的支持并发，也可以在此引入Active Object甚至Leader/Followers。

同样提供了针对REST的端口，对应适配器为Controller。它主要是为了支持移动终端设备、Web应用，以便于相关人员直接发出发货或收货请求。

同样提供了DB的端口。这个数据库是对应仓库的专有数据库，与库存管理数据库无关。

提供了针对设备（指仓库的设备，如叉车，箱子，运输车等）的端口，对应适配器为South Gateway。

提供了针对配置文件的端口，对应适配器为Configurer。此功能是为了支持拓扑信息的动态配置。

提供了针对外部物流系统的端口，这里为其建立了Shipping的防腐层，使其能够更好地支持各个不同的物流供应商。

目前，我针对可视化架构与设计的手段仍在完善之中，并已经尝试在真实项目中实践以进行验证，并希望能够找到足够简单的方法，为架构师与开发者提供直观而又具有体验价值的沟通方式，并能形成行之有效的设计手段。

张逸，现就职于ThoughtWorks中国。作为一名咨询师，主要为客户提供组织的敏捷转型、过程改进、企业系统架构、领域驱动设计、大数据、代码质量提升、测试驱动开发等咨询与培训工作。

阅读原文