研发效能事业部
Alibaba R&D Efficiency

VOL.01

研发效能走进阿里

领域驱动设计的理论与实践

——如何高效准确地发现和保持领域模型

钉钉扫码签到

钉钉扫码入群

领域驱动设计的理论与实践

如何高效准确的发现和保持领域模型

张刚 博士

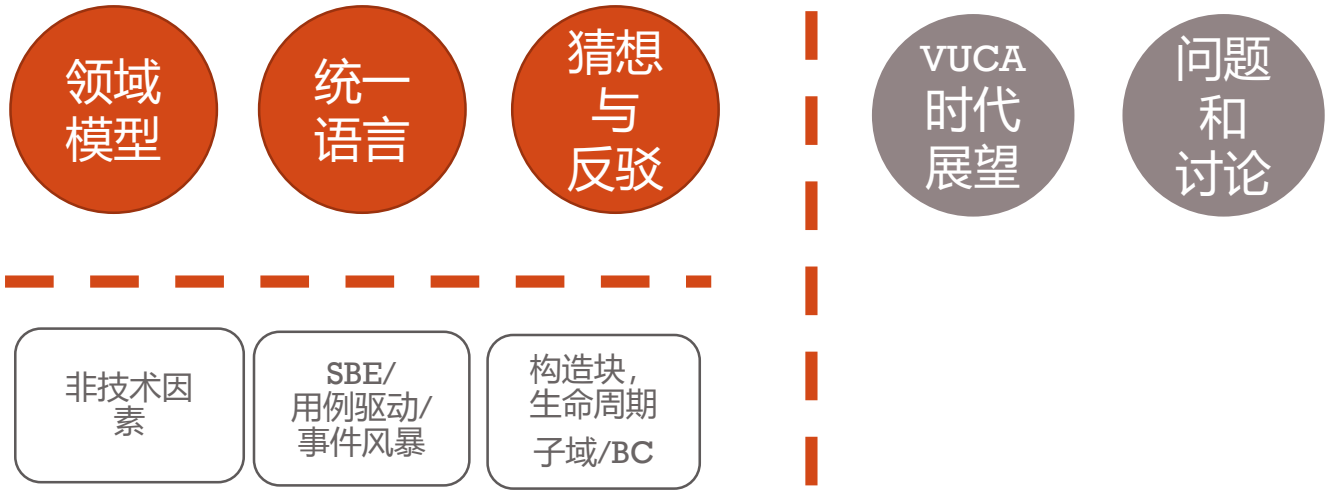
张刚

- 一线实践者，写了18年代码，希望通过持续的方法演进，提高行业的软件生产力水平
- 领域建模上的经验
  - 2005年统一过程（RUP），接触领域建模方法
  - 2009年领域工程、精益和敏捷方法
  - 2011年领域驱动设计
  - 在N个自有产品及M个咨询项目上实践过领域建模及相关实践，收到了不错的效果
- 曾经是某电信产品架构团队负责人，创业公司CTO和技术合伙人、软件架构顾问以及高校教师

关于话题

领域驱动设计的理论与实践

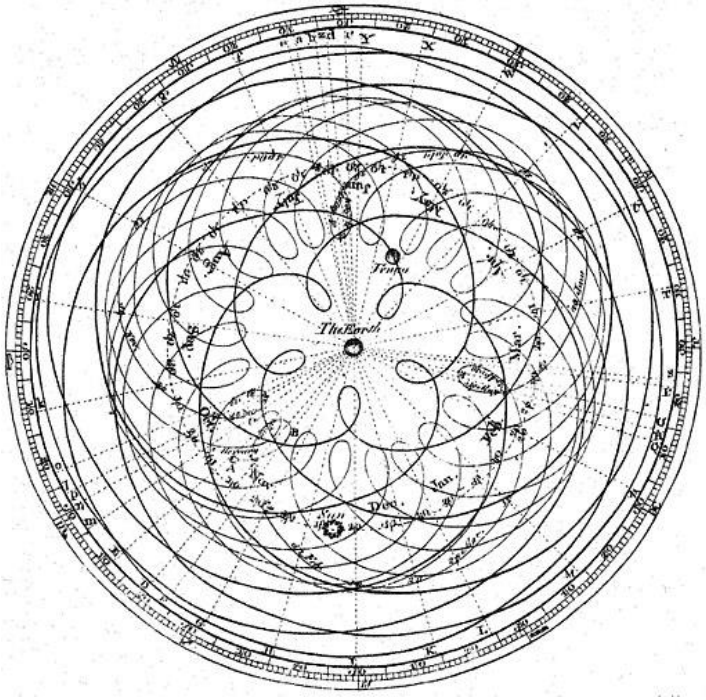
如何高效准确的发现和保持领域模型



我们没有做领域建模，软件也开发出来了

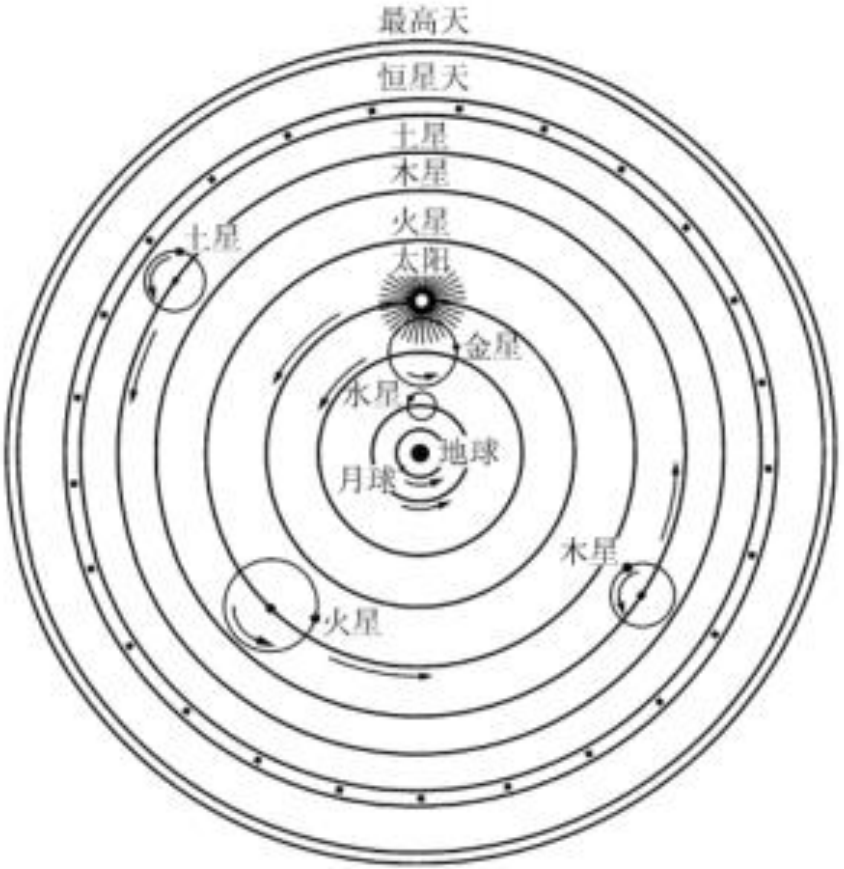
只是后面增加功能的时候，遇到了不少麻烦

1 模型的本质



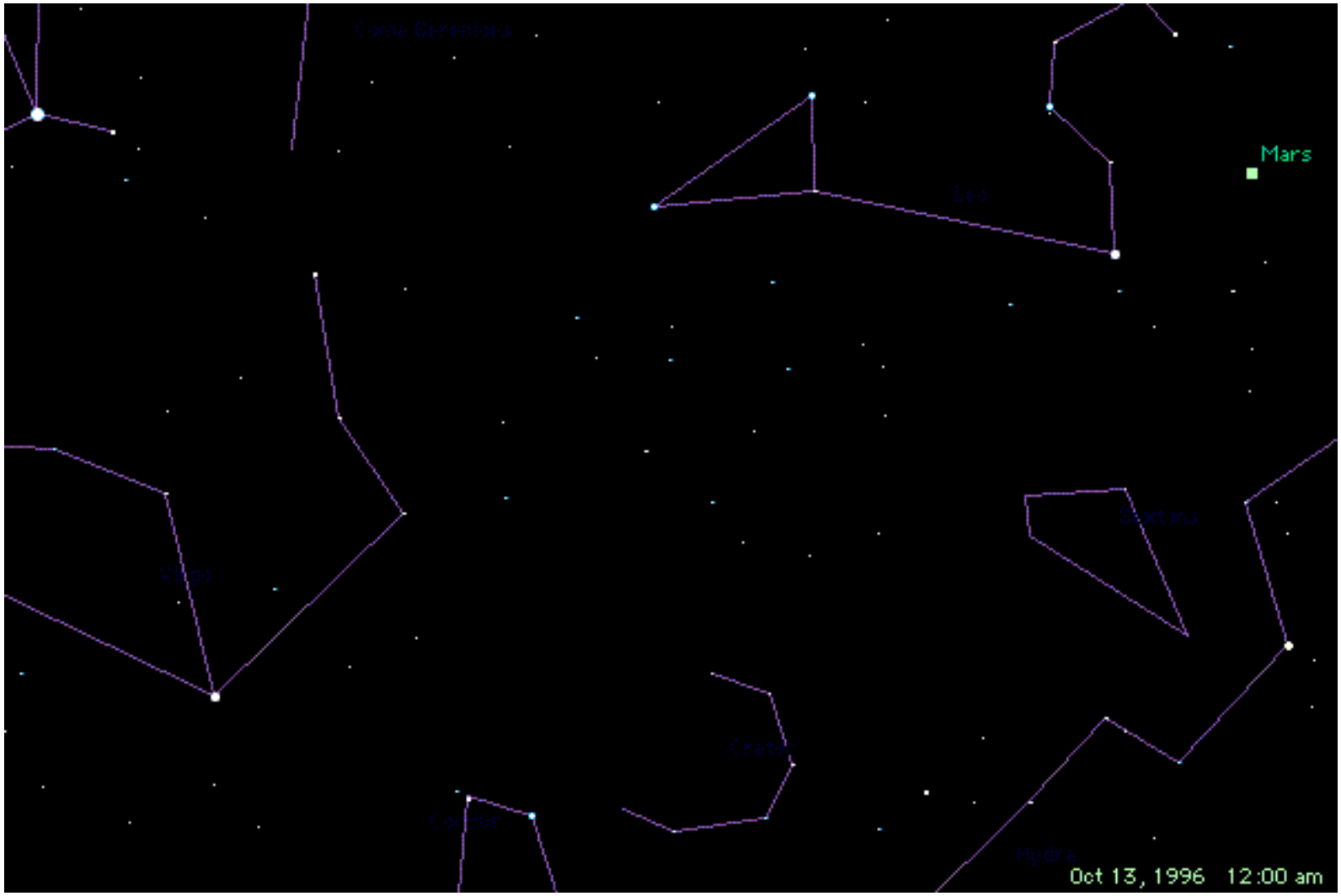
模型的本质—我们认识世界的方式

太阳东升西落（现象）

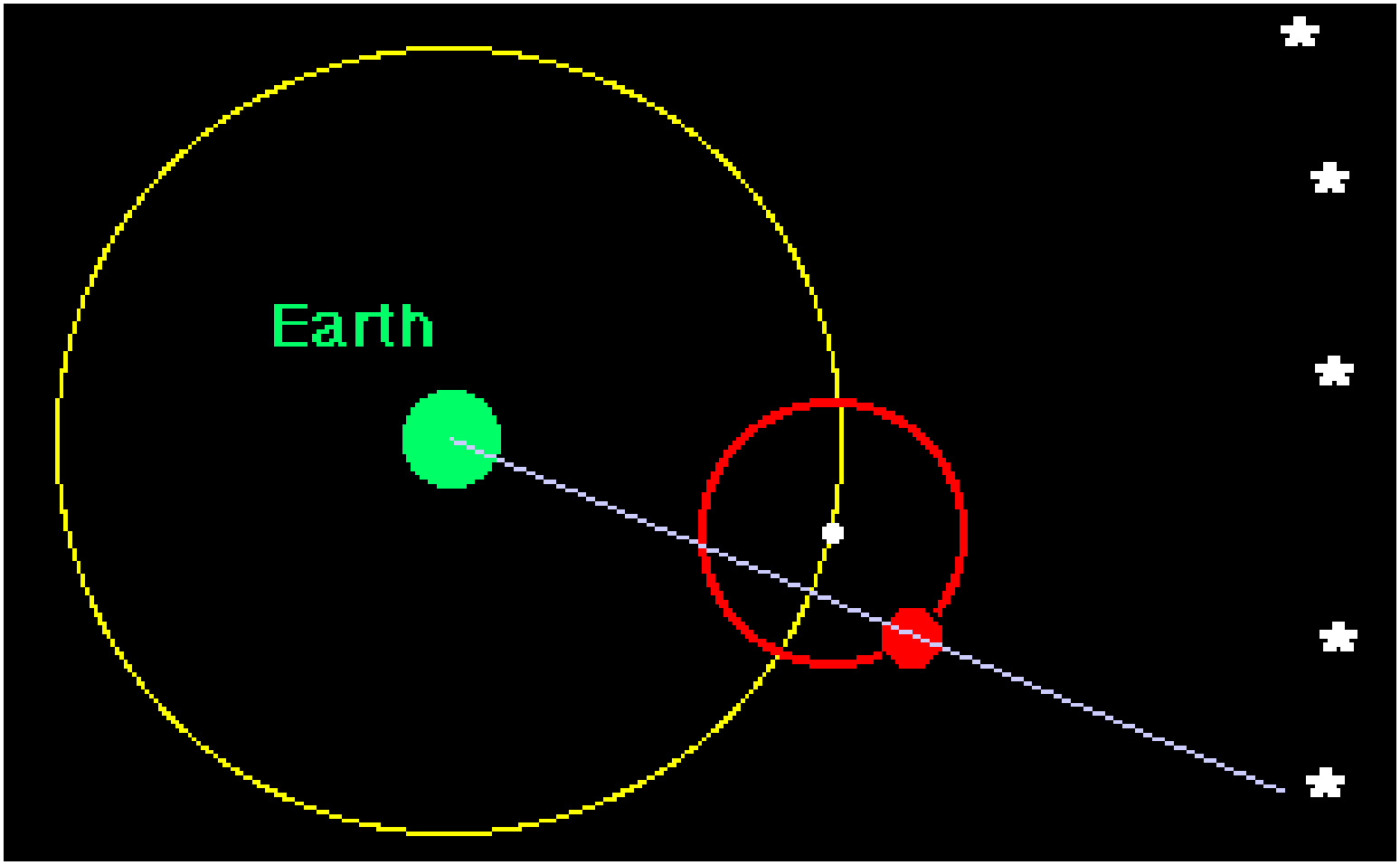


（模型）

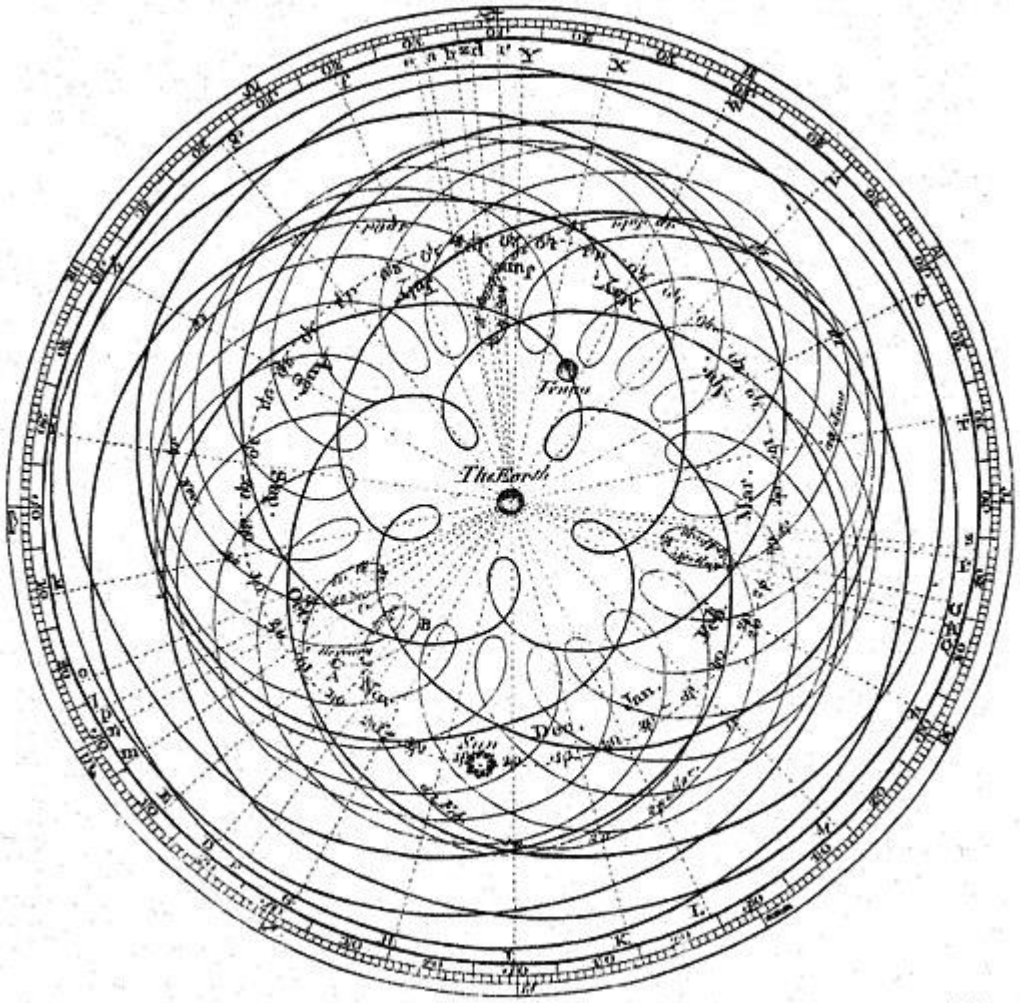
模型和观测产生矛盾



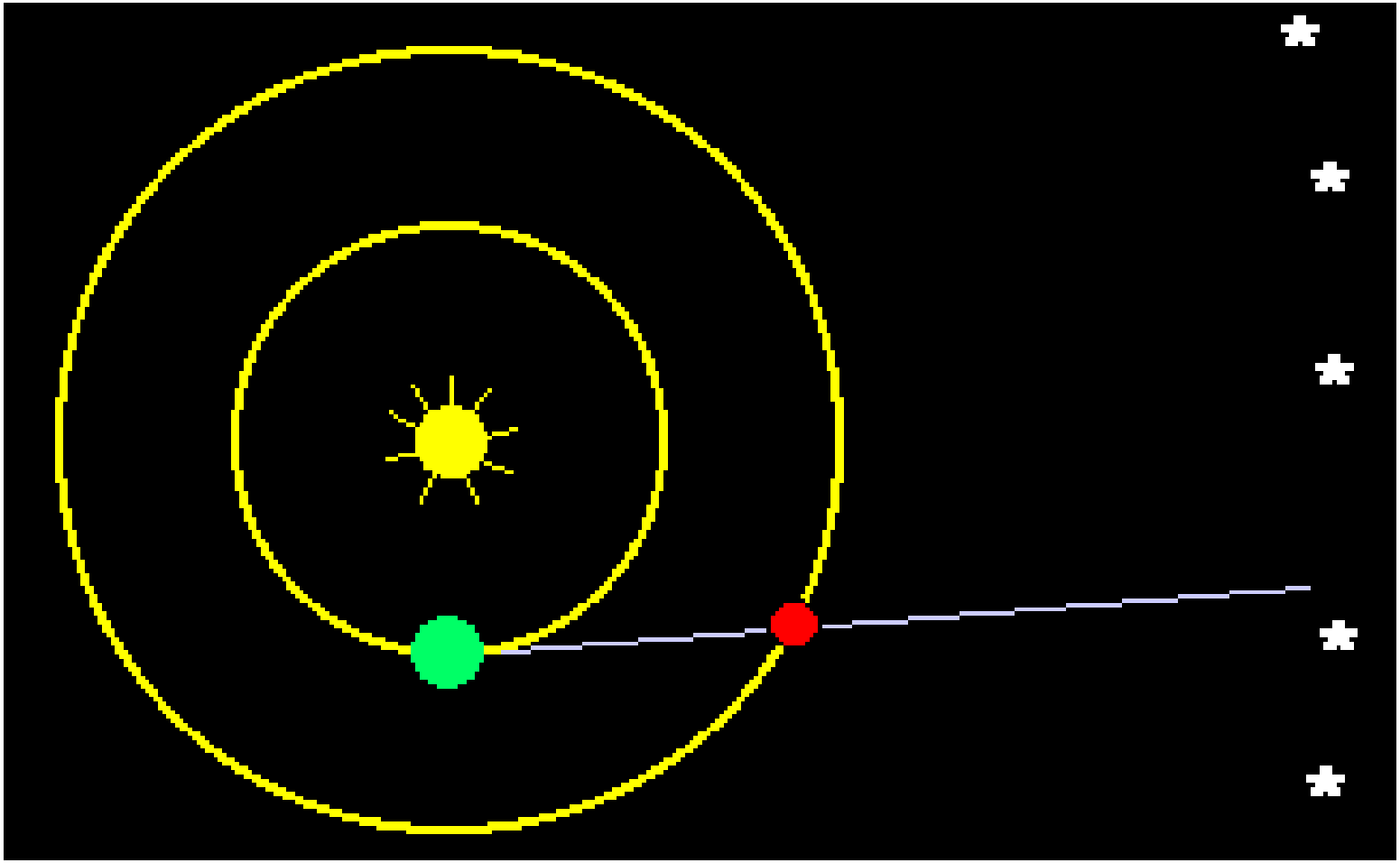
本轮-均轮模型



修补，或者突破



修补，或者突破



启示1：

功能只是表象，模型才是内在

好的模型，可以让功能实现变得更容易

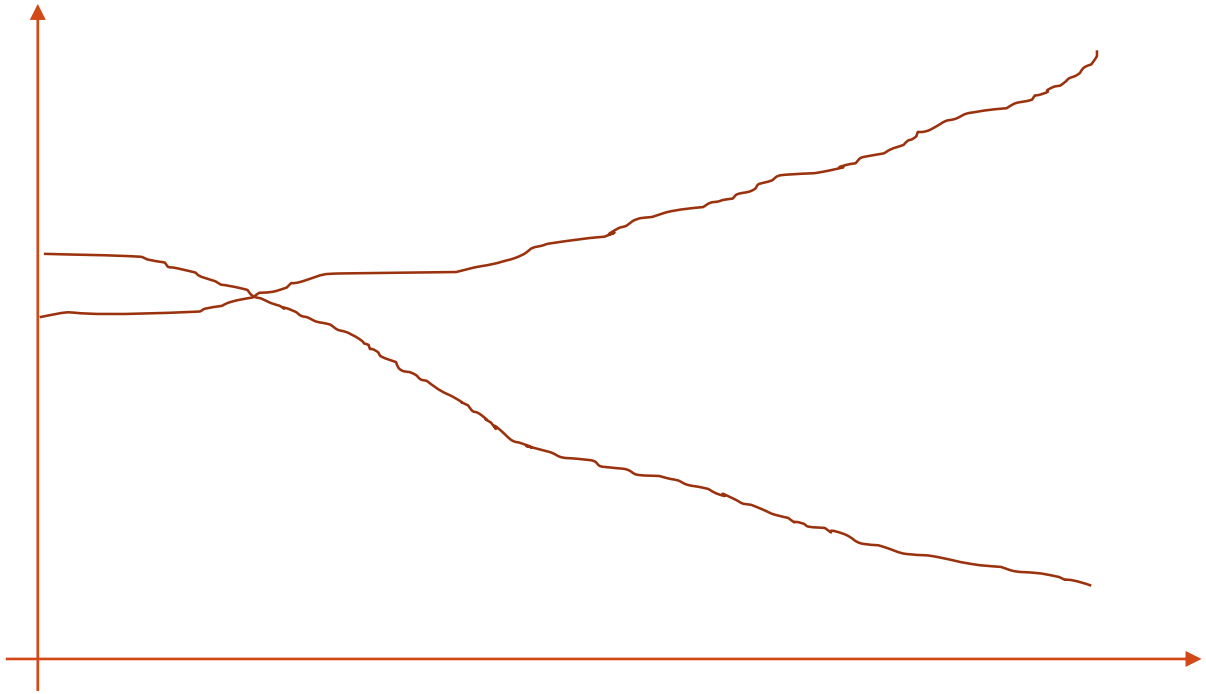
我们没有做领域建模，软件也开发出来了

只是后面增加功能的时候，遇到了不少麻烦

一个定性分析

资产还是负担？

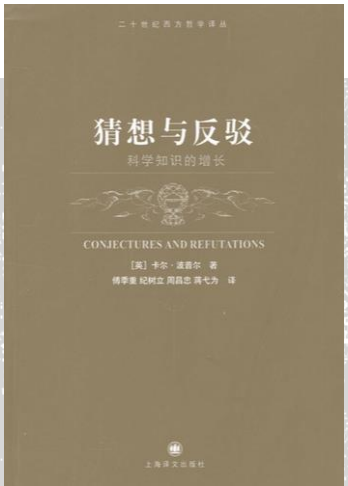
- 既有资产，可能会成为实现新功能的可复用资产（好的模型）
- 既有资产，也可能会成为实现新功能的约束（不好的模型或者没有模型）



小结

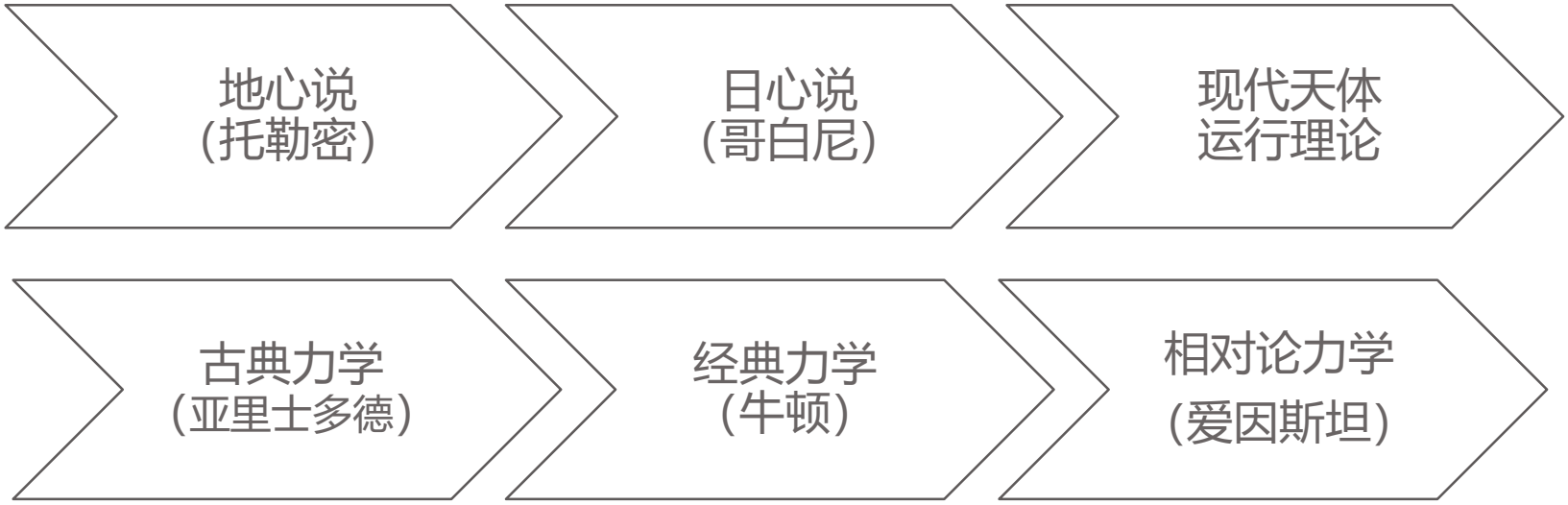
- 功能只是表象，模型才是内在
- 好的模型，可以积淀为组织的资产
- 不好的模型，可以积淀为组织的债务

2 从认知角度看领域建模



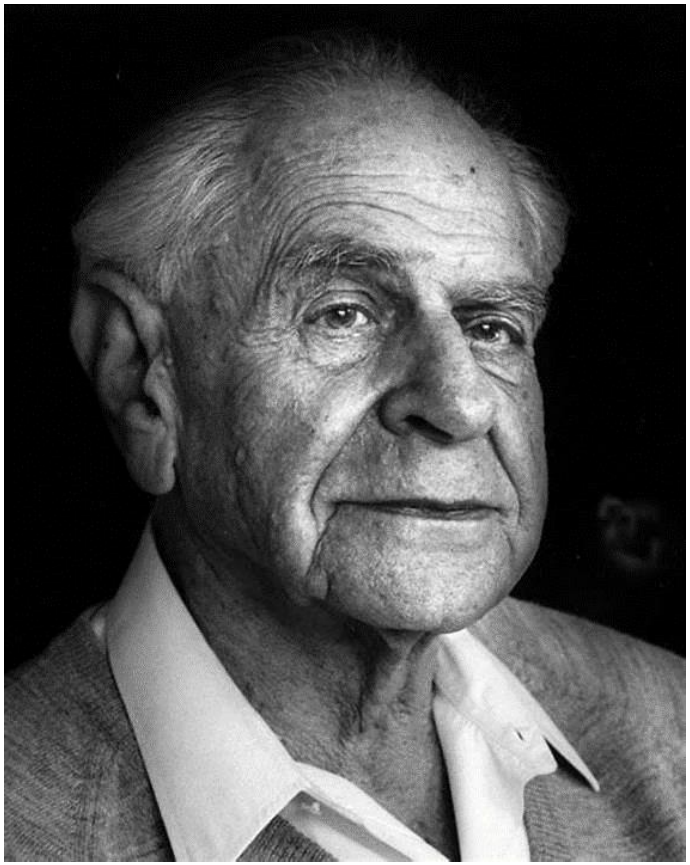
地心说在今天看起来失败了，

但是它真的是一个荒谬的理论吗？

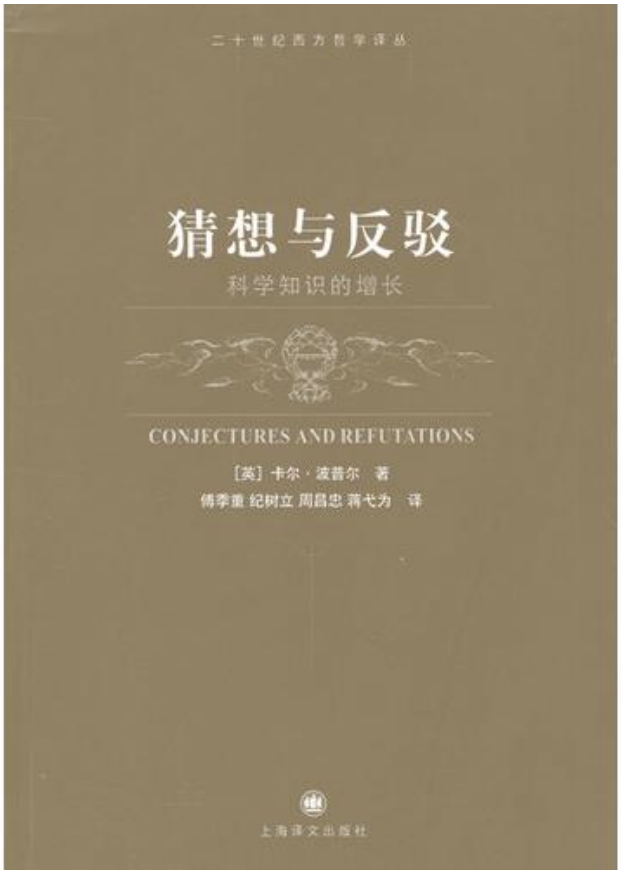


科学发现，是一个不断否定、逐步求精的过程

猜想与反驳-科学发现的基本范式



卡尔·波普尔（Karl Popper）
1902-1994



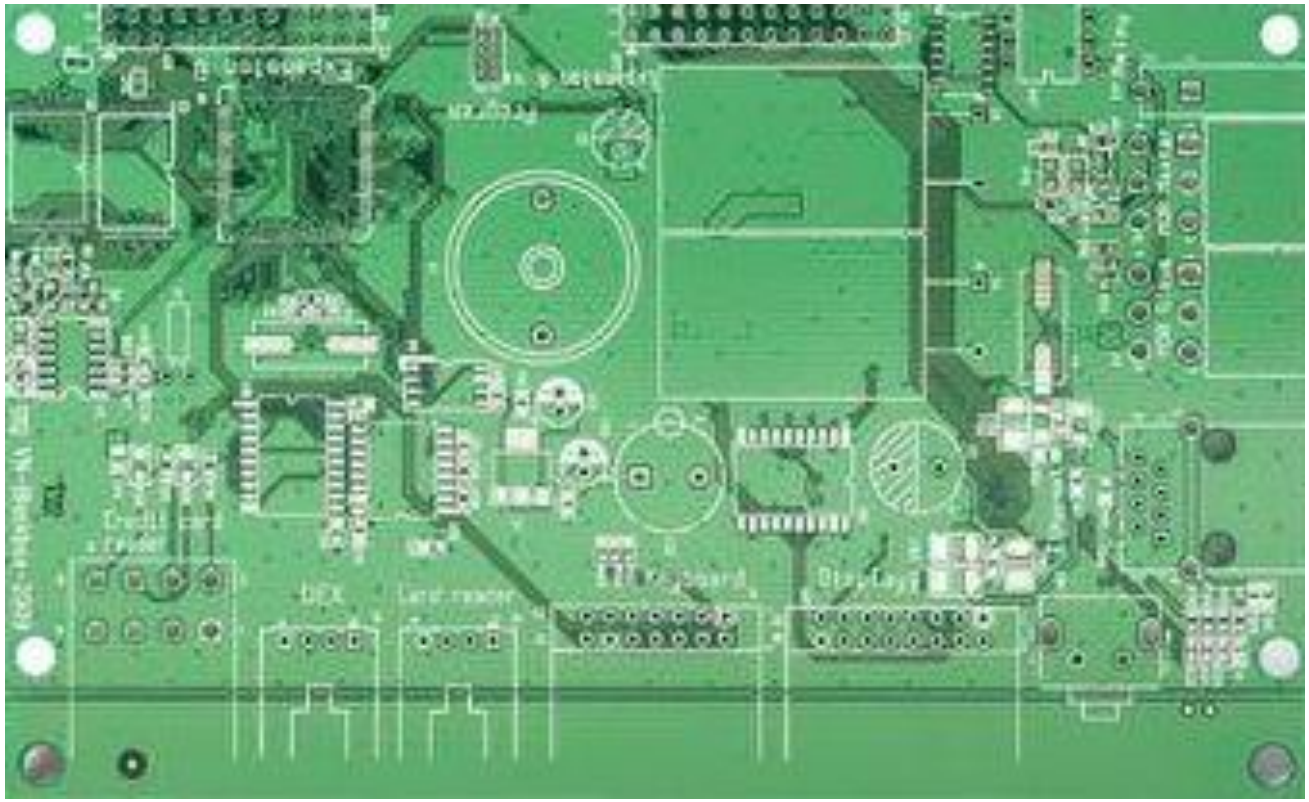
科学知识的可错性（可证伪）
科学发现是猜想与反驳的过程

软件建模过程的猜想与反驳

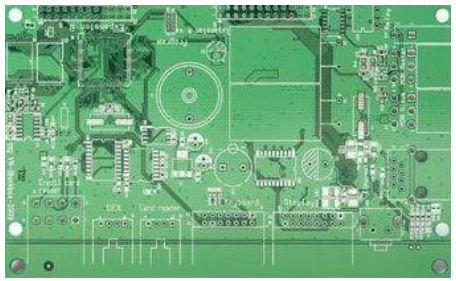
案例来源：

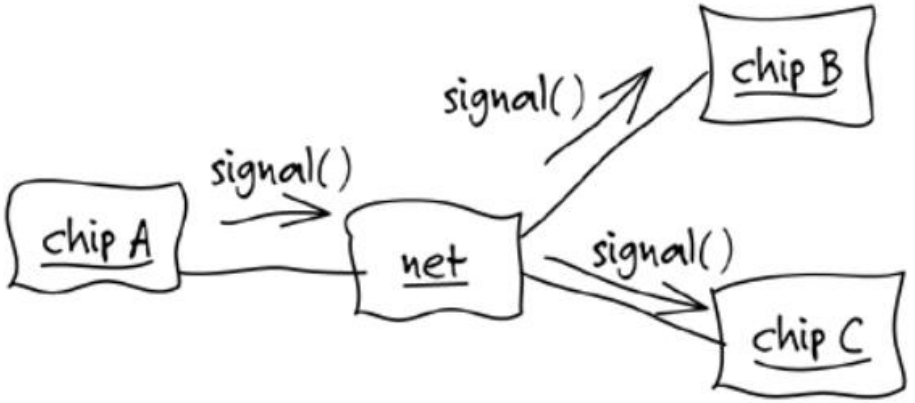
Eric Evans《领域驱动设计》

任务-开发一个EDA系统



Net是一种可以连接任意Chip的导线





电信号在芯片之间的流动关系

讨论

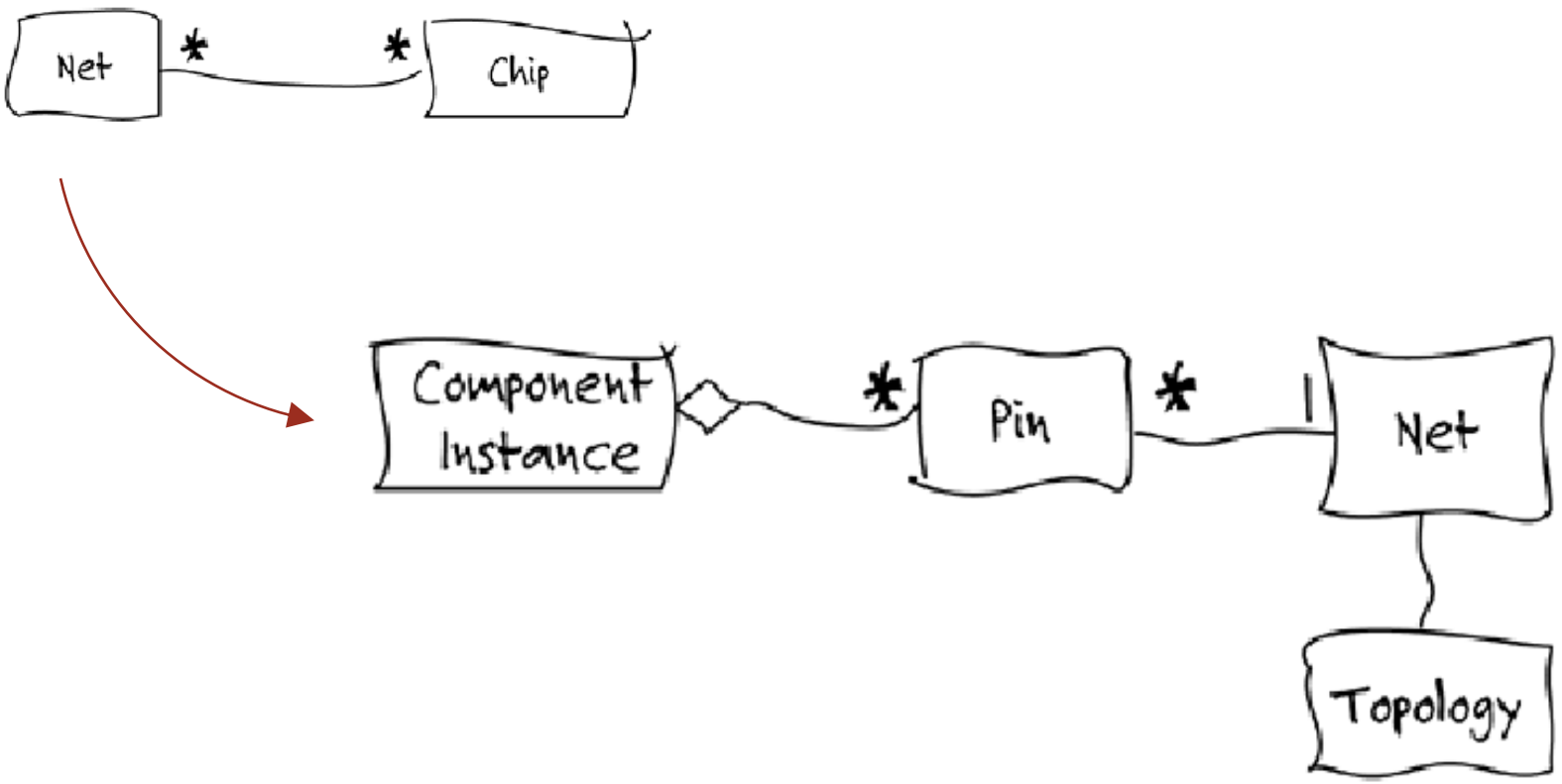
元件不一定是芯片。
…
我们一般称它们为Component Instance

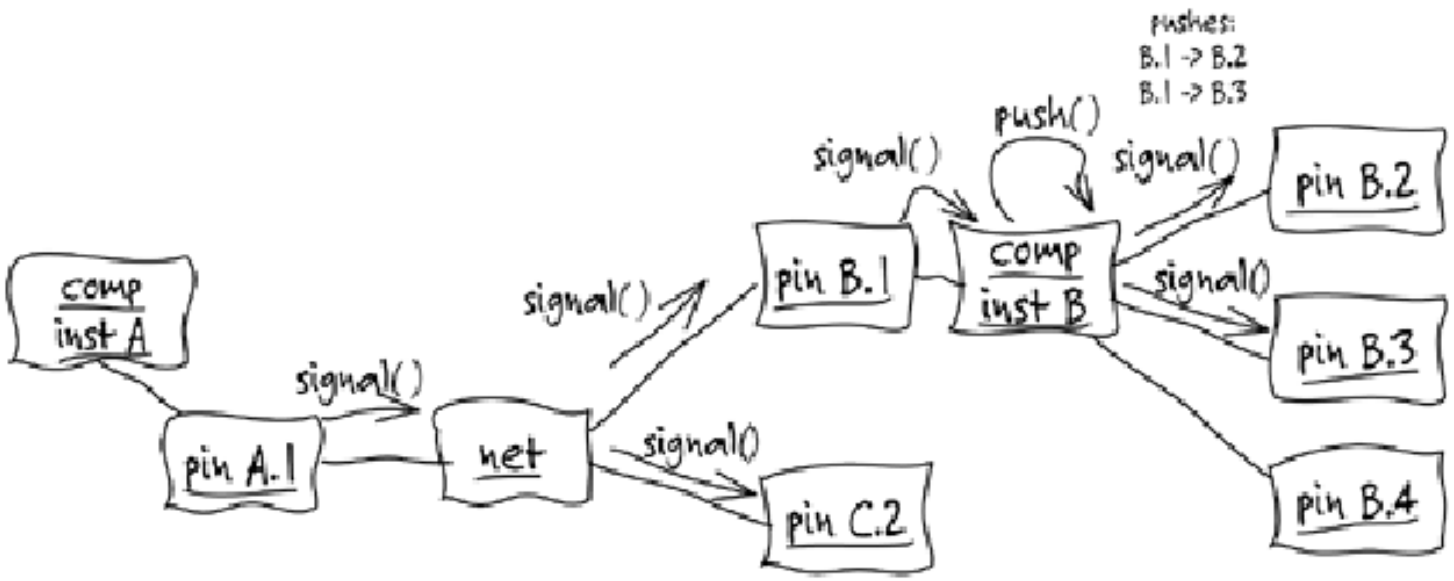
仅仅说一个signal到达ref-des是不够的

..

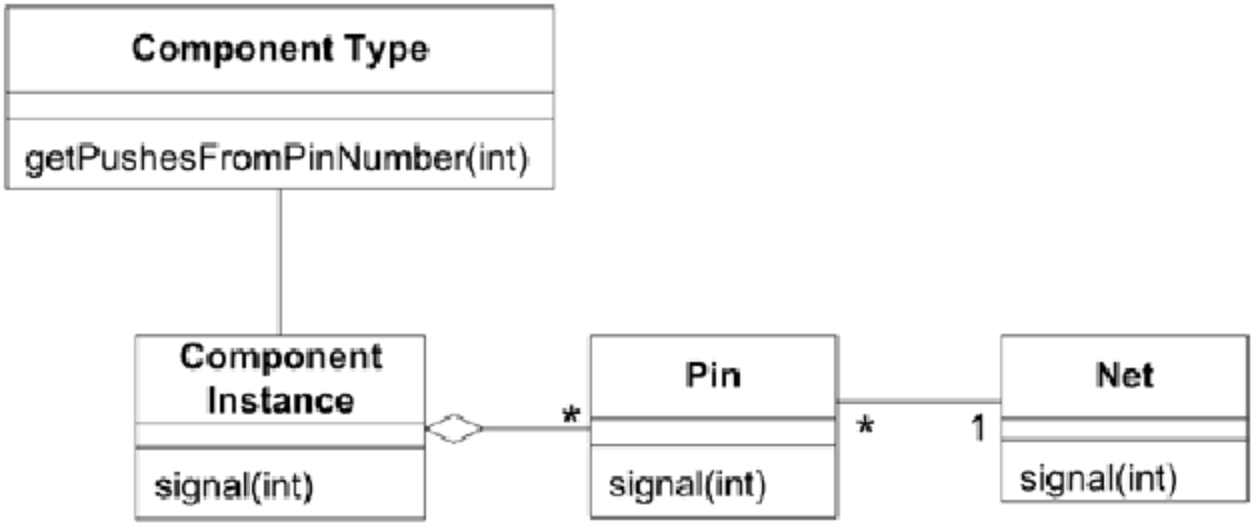
..

net将一个实例的某个引脚和另一个实例的引脚相连





模型



启示2：

猜想与反驳是科学发现的基本特征

建模过程是不断猜想与反驳的过程

演化的观点是建模过程的基本心智模式

领域模型的本质就是知识

- “有用的模型很少停留在表面层次上。随着对领域和应用的理解逐步加深，我们往往会丢掉那些最初看起来很重要的表面元素，或者切换他们的角度。
  这样，一些在开始时不可能发现的巧妙抽象就会浮出水面，而他们恰恰切中问题的要害”

Eric Evans

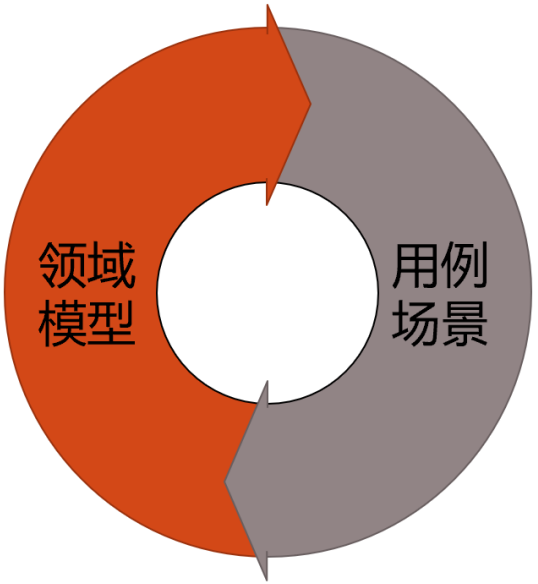
小结

- 建模过程是一个猜想和反驳的过程
- 推论：
  - 领域建模发生在业务领域
  - 领域建模需要持续进行

3 统一语言和协作

业务、开发、测试紧密配合，

在澄清需求的过程中逐步完善领域模型

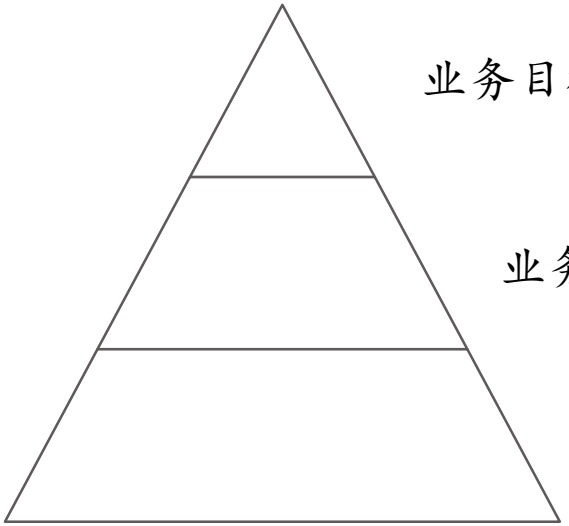


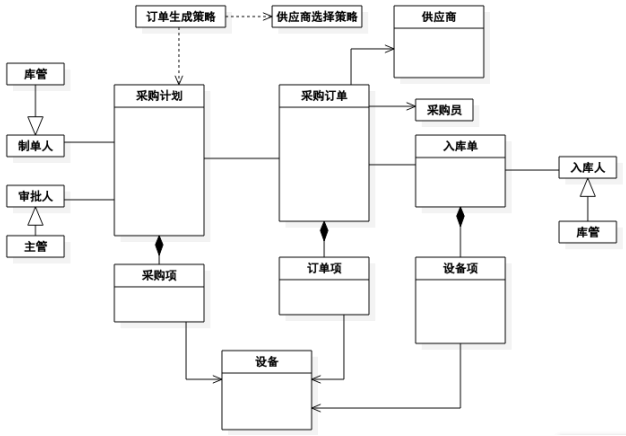
我想做领域建模，但是我们的业务人员不重视……

用共同的价值来推动

左手栏

右手栏





让讨论聚焦于此

让概念收敛于此

如何保证领域模型和需求严格同步？

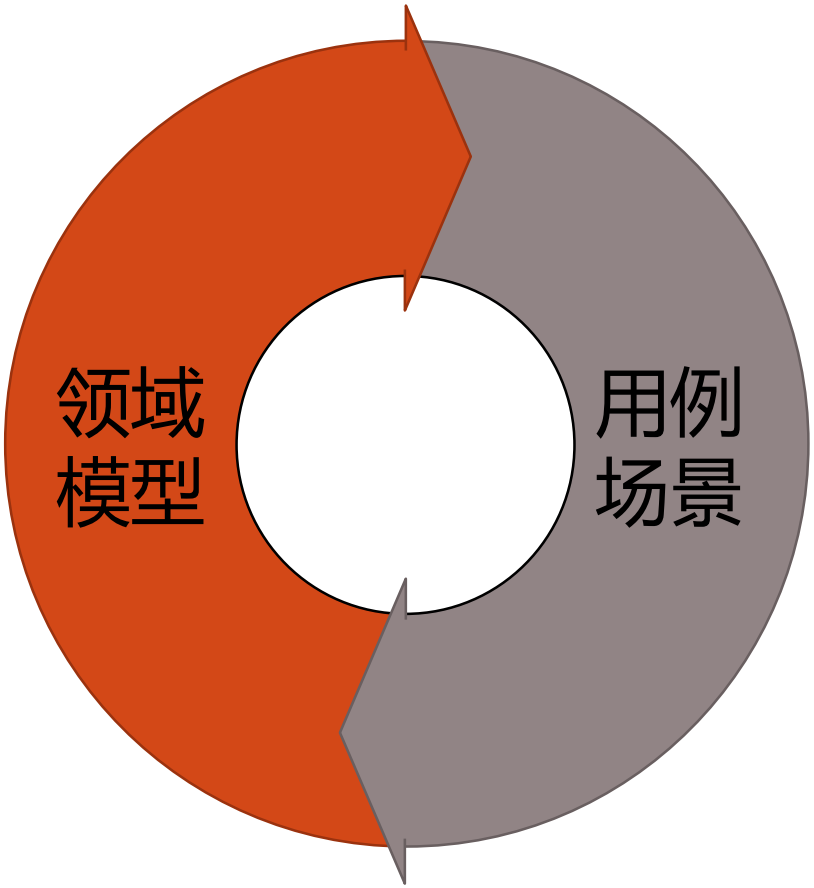
- 一个简单规则：任何在需求描述中出现的概念，都必须出现在领域模型中。如果需求描述中存在概念之间的关系，领域模型中也必须有这个关系。

统一语言

- 将模型作为语言的中心，确保团队在所有的交流活动和代码中坚持使用这种语言。在画图、写文档和说话时也采用这种语言。
- 试着尝试修改模型以及对应的语言表达来消除不自然的地方（相应的也要修改代码，包括模块名、类名、方法等）。

通过猜想，建立关于领域模型的概念假设

使用用例场景进行反驳，促进领域模型逐步求精



一个示例：

- 初始需求：设备采购和管理

设备管理部门负责制定和提报采购计划，并提交上级部门批准。批准后系统将生成采购订单。采购部分负责根据订单采购。设备采购到货后，仓库管理人员负责验货，验货合格后入库。

提报采购计划

采购计划包含什么内容？

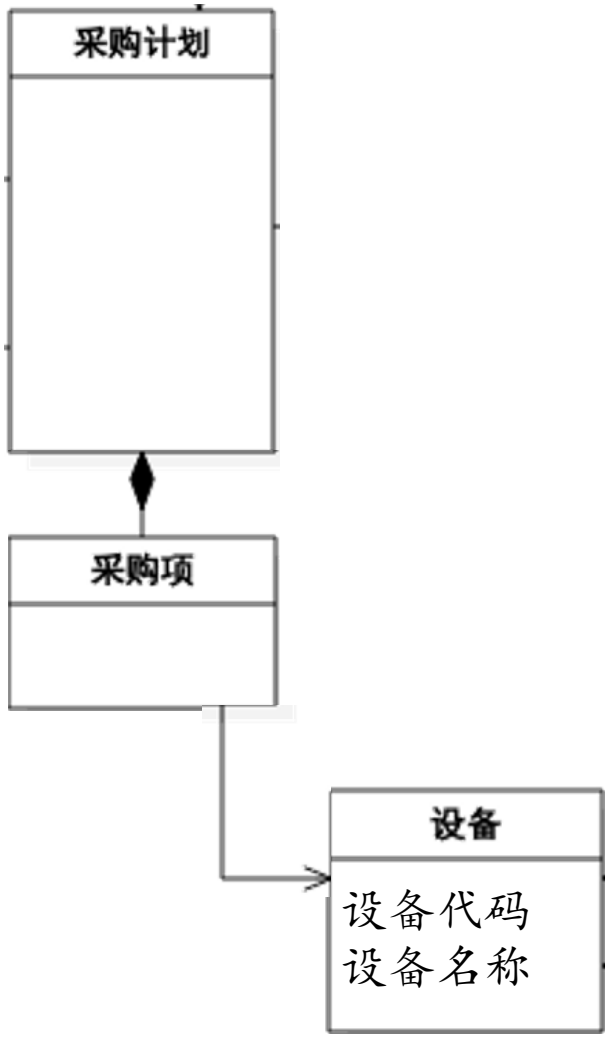
设备代码、设备名称、数量、制单人、制单时间…

设备名称和设备数量的关系看起来更近，它们可以分为一组

一个采购计划仅仅包含一种设备吗？

不是。我希望一个采购计划能包含多种设备。

那看起来我们需要增加一个采购项的概念。



提报采购计划

数量、制单人、制单时间…

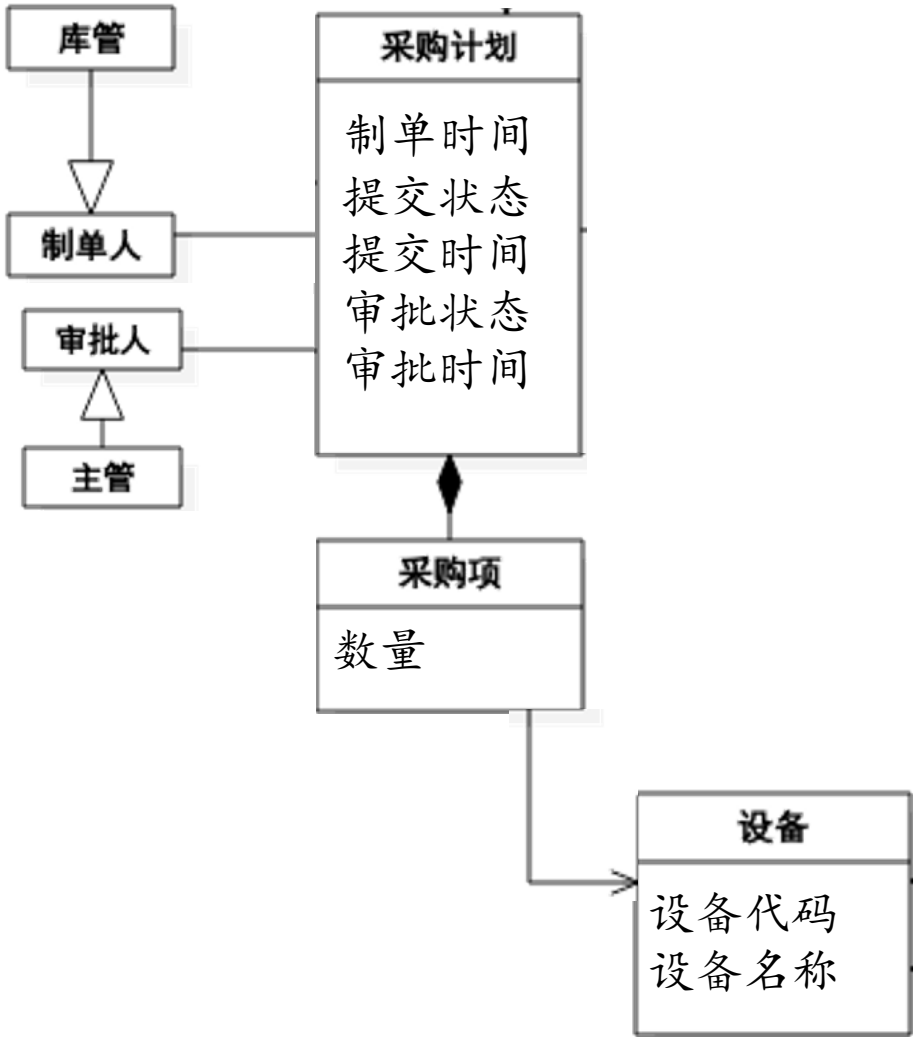
制单人看起来是一种角色。它目前由库管来扮演。

只有库管才会提报采购计划吗？

暂时是这样。

那好

对了， 后续我们可能还需要记录其他状态，比如提交状态、提交时间、审批状态、审批人、审批时间等



生成采购订单

是一张申请单就对应一张采购订单吗？

不一定。可能我们需要根据供应商来进行拆分

哦，你提到了一个新的概念，供应商。让我们把它记下来。

那么，申请的时候没有供应商，我应该怎么选择供应商呢？

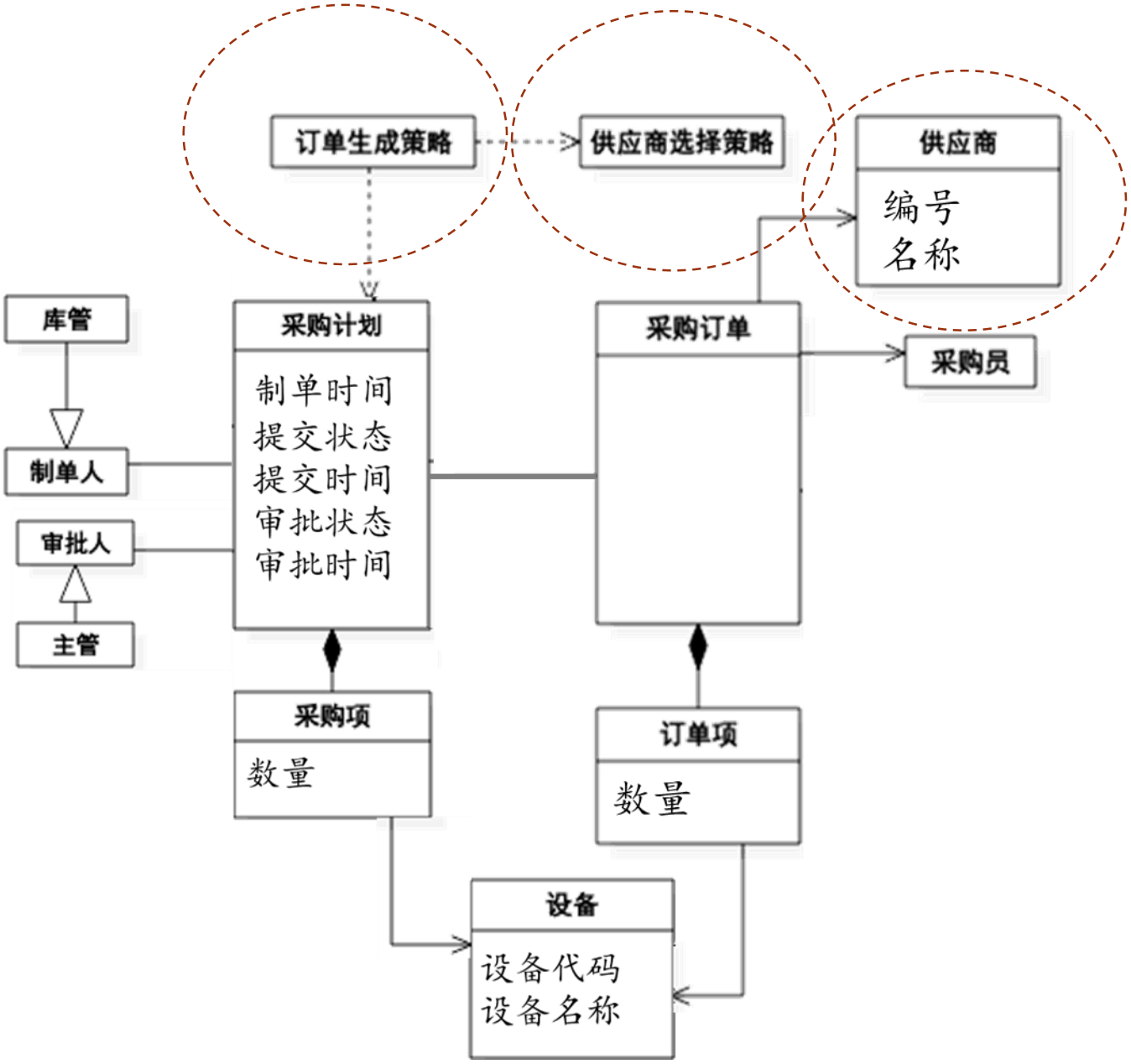
这个过去好像没分析过...

我们的系统中应该建立一个供应商管理的模块，每种设备都有它的默认供应商。

或者是我们的供应商有一个评级，只要这个供应商能提供这种设备，我们就总是从高优先级的供应商选择

我们发现了一个新的需求，供应商管理

供应商管理看起来是一个复杂的需求，让我们下一次再讨论。目前让我们把这个规则暂时记为“供应商选择策略”。在确定下来之前，我们团队将使用最简单的规则来开发。也就是，我们系统中仅有一家供应商。当然这只是一个假设。



设备入库

每张订单一定是同时到货、同时入库吗？

不一定。同一张订单的到货时间可能不一样。我们希望在到货后就立即入库。

那我们需要创建一个新的概念，叫做入库单。

嗯，我们似乎有一个新的需求，跟踪订单的状态，以及订单是否已经全部完成。

好的，让我们把它记下来。

现在让我们来分析一下入库单有哪些内容

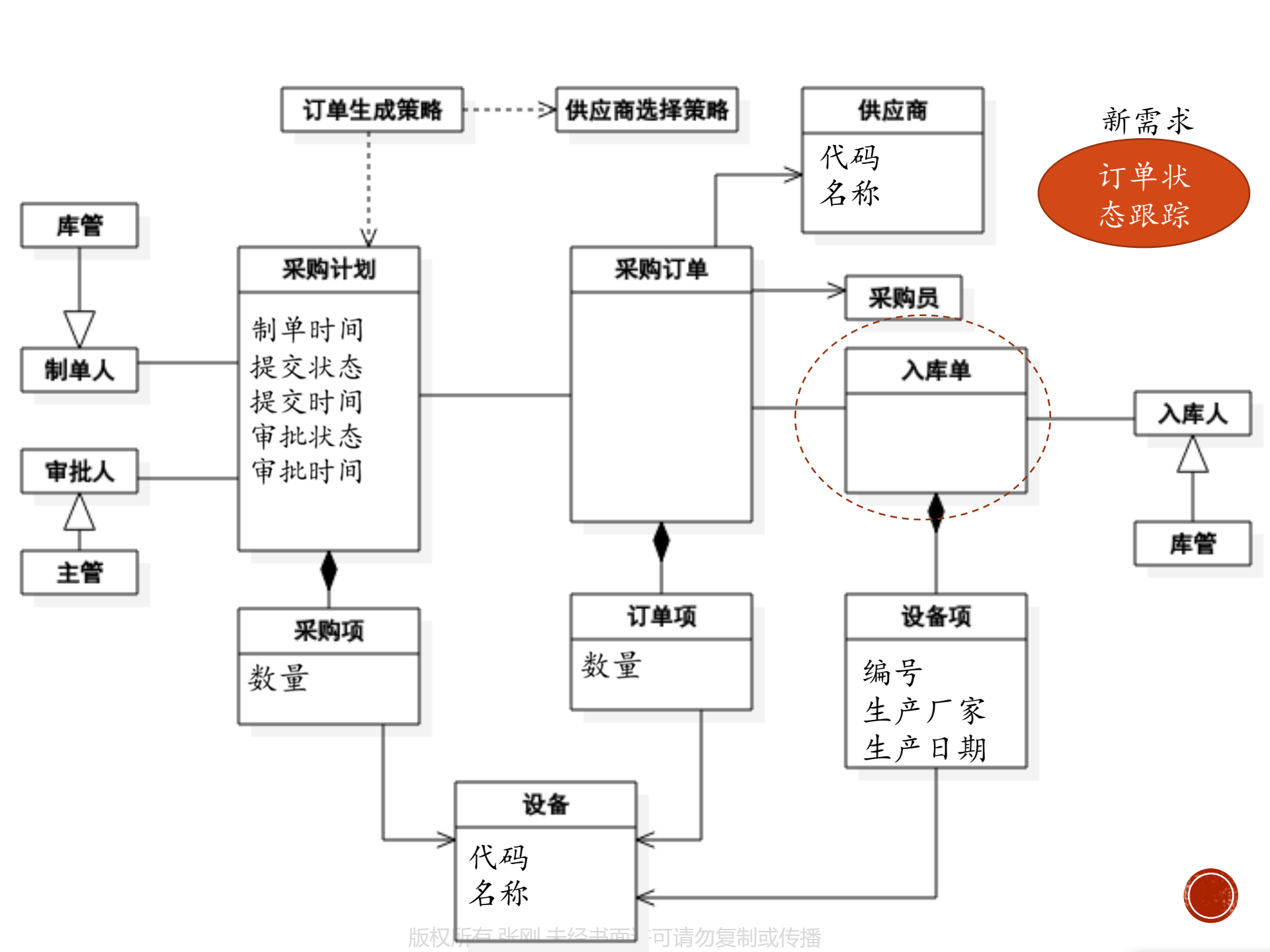
我希望它包括采购订单号、供应商编号、供应商名称、设备编号、设备代码、设备名称、设备型号、设备规格、生产厂家、生产日期、出厂日期、收货人、收货时间等等

好的，供应商这些信息是在订单里面已经包括了，这些我们只要保持一个入库单到订单的关系就有了。

是这样。

但是，设备编号和设备代码是两个概念对吧，这是说我们的仓库是对每一个设备都单独管理的，是这样吗？

好的，让我们明确区分采购申请中的“设备”和现在的“设备”。这个让我们和它叫设备项吧。



设备入库

我发现设备里面出现了两个新的概念，设备型号和规格。它们是在什么时候确定的？

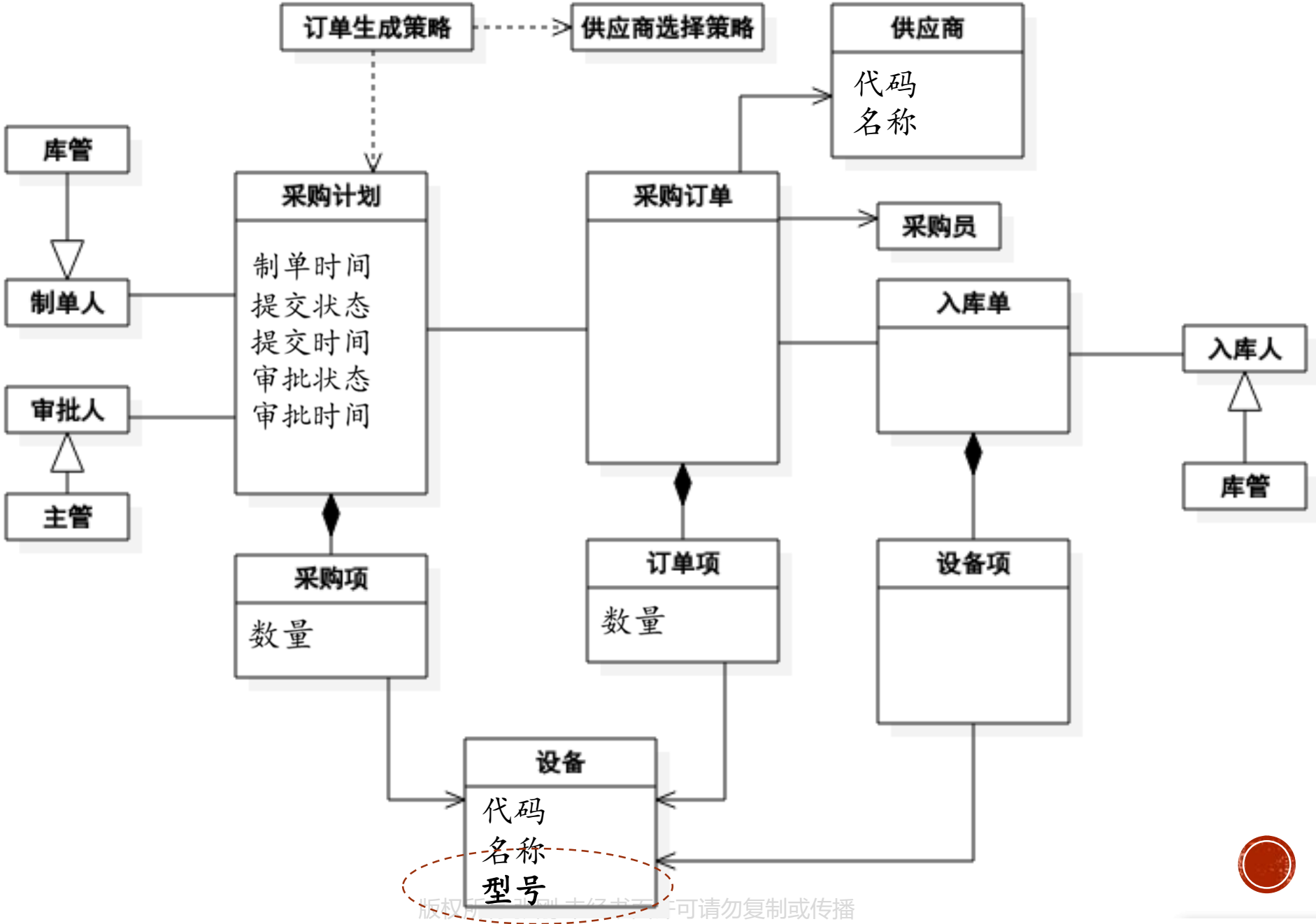
嗯，这里是一个遗漏，如果一个设备存在多种型号规格，应该在采购申请的时候就指定。

好的，让我们把这个概念补入到采购申请里面。

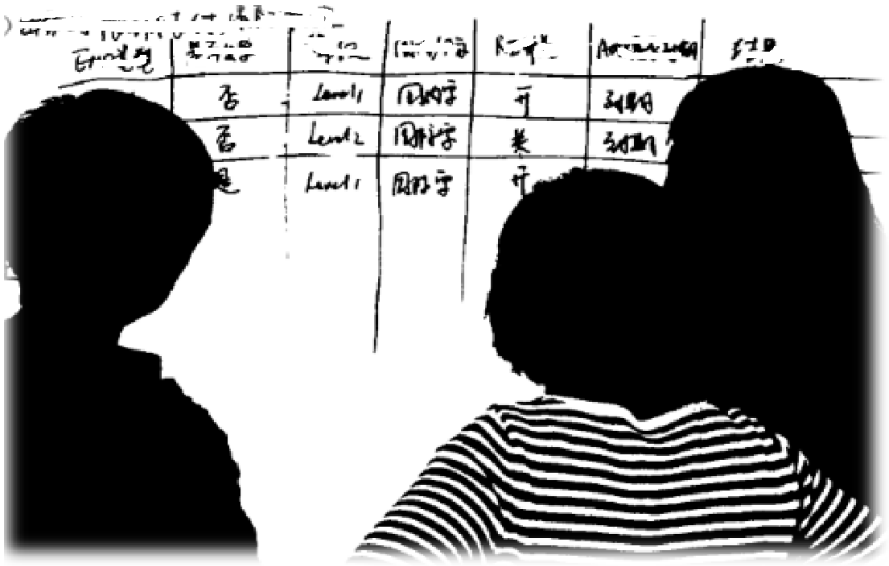
对了，什么叫型号，什么叫规格？你能举一个例子吗？

这两个概念确实有些差别……
不过，为了让系统容易使用，让我们把它们统称为”型号”吧。

好的，让我们用型号来指称这个概念。



实例化需求和领域建模配合紧密



业务规则和测试示例
都应该使用领域模型

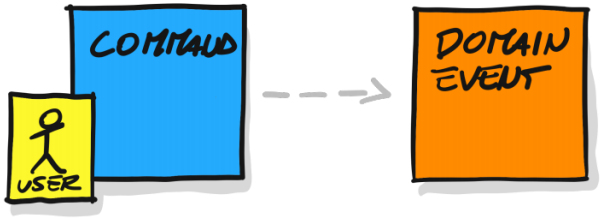
小结

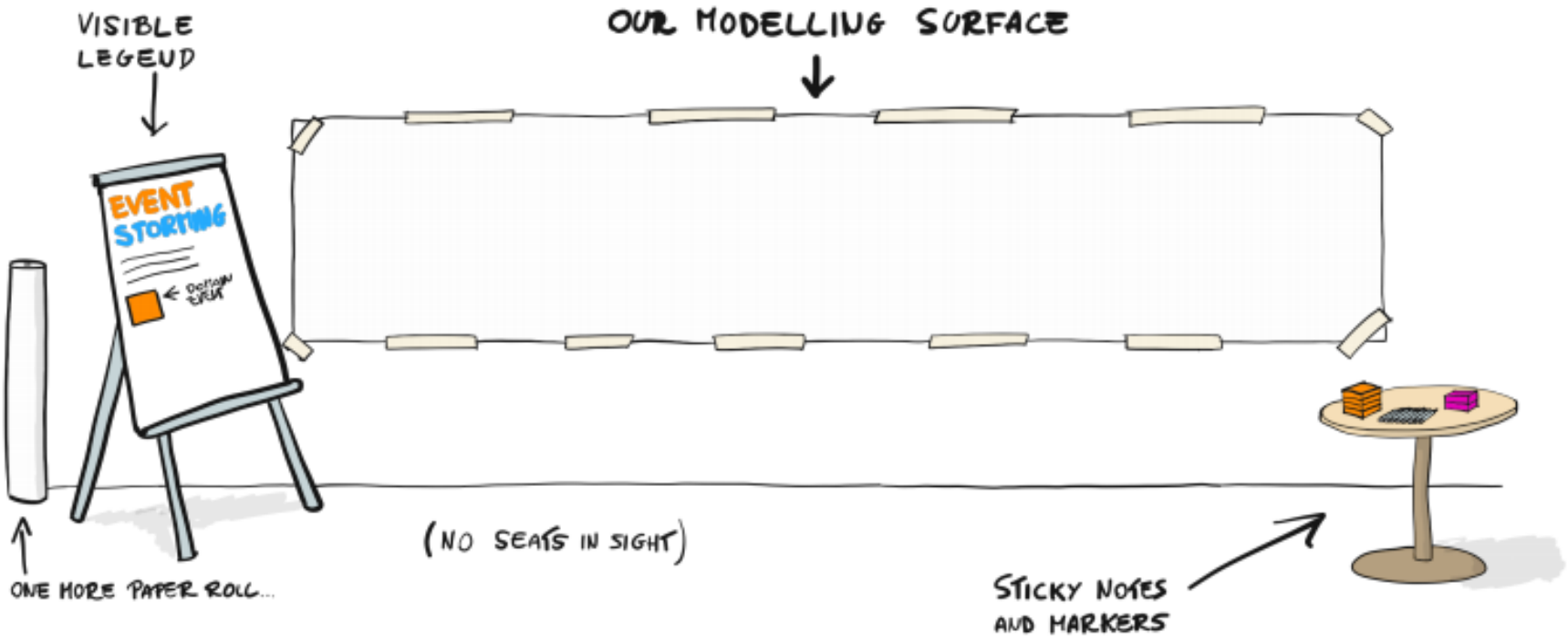
- 写下来的领域模型，可以澄清不同参与者的内心假设，发现潜在的概念冲突和不一致的地方；
- 对于领域模型中数据来源进行分析，可以发现缺失的用户场景。
- 用领域模型作为统一语言，强迫团队在沟通需求时使用领域模型，是确保领域模型始终有效且同步演化的方式。

事件风暴和领域建模

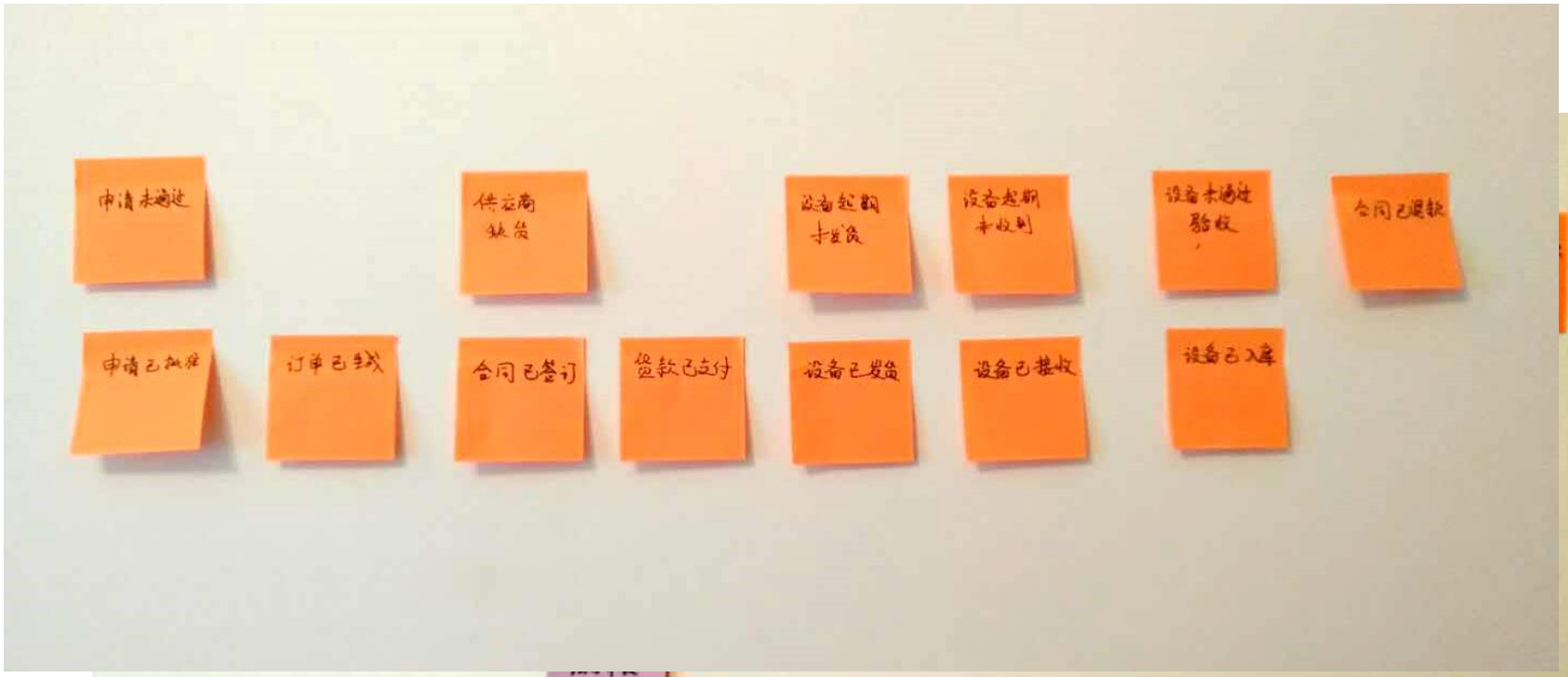
用事件风暴来发现和精化业务场景

Event Storming





By Alberto Brandolini

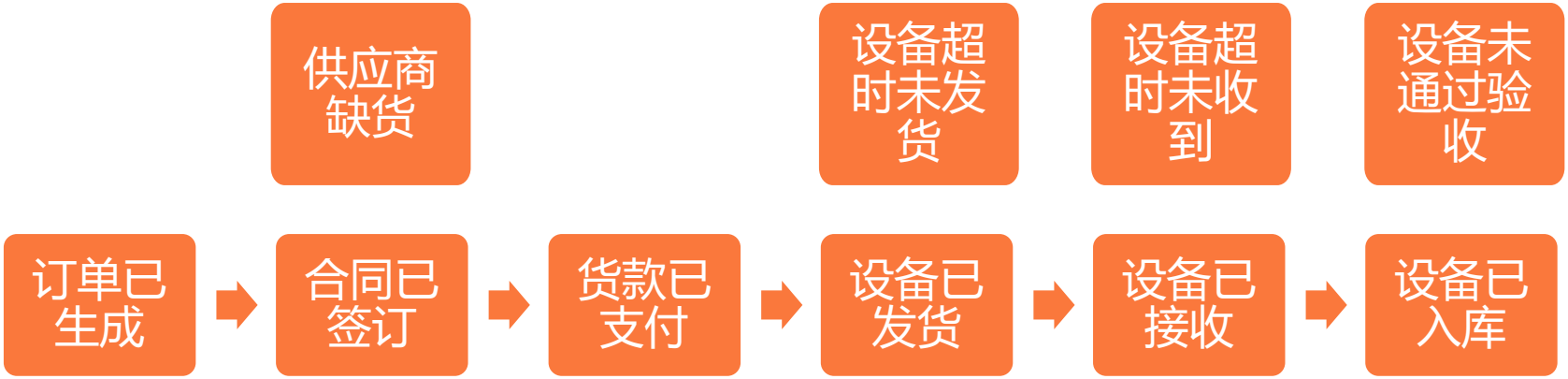

申请未通过
供应商
缺货
设备起期
未发货
设备起期
未收到
设备未通过
验收
合同已退款
申请已批准
订单已生成
合同已签订
货款已支付
设备已发货
设备已接收
设备已入库

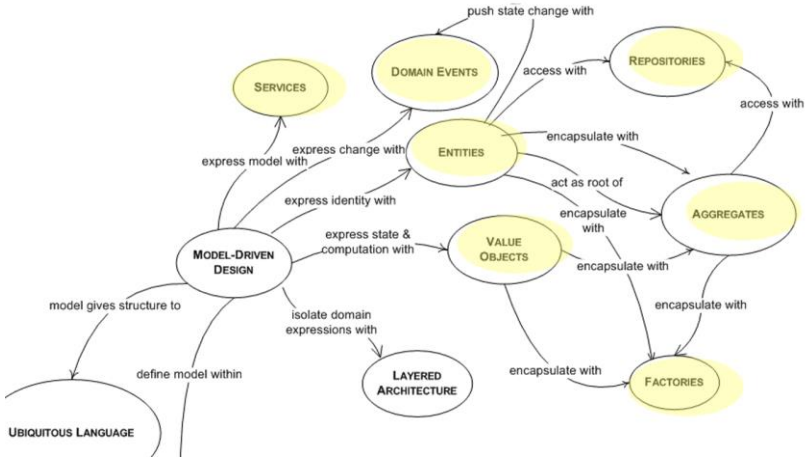
事件风暴的典型实践

- 允许许多参与者同时参加
- 以领域事件作为领域分析的核心
- 使用事件(Event）、命令（Command）、参与者（Actor）等作为建模的基本元素
- 把元素以聚合为核心进行组织

事件风暴的最有价值的部分

- 从价值角度来审视业务流程的合理性
  - 领域事件容易创建业务人员和非业务人员的共识
  - 由后至前的分析，聚焦价值有助于发现业务流程中的遗漏

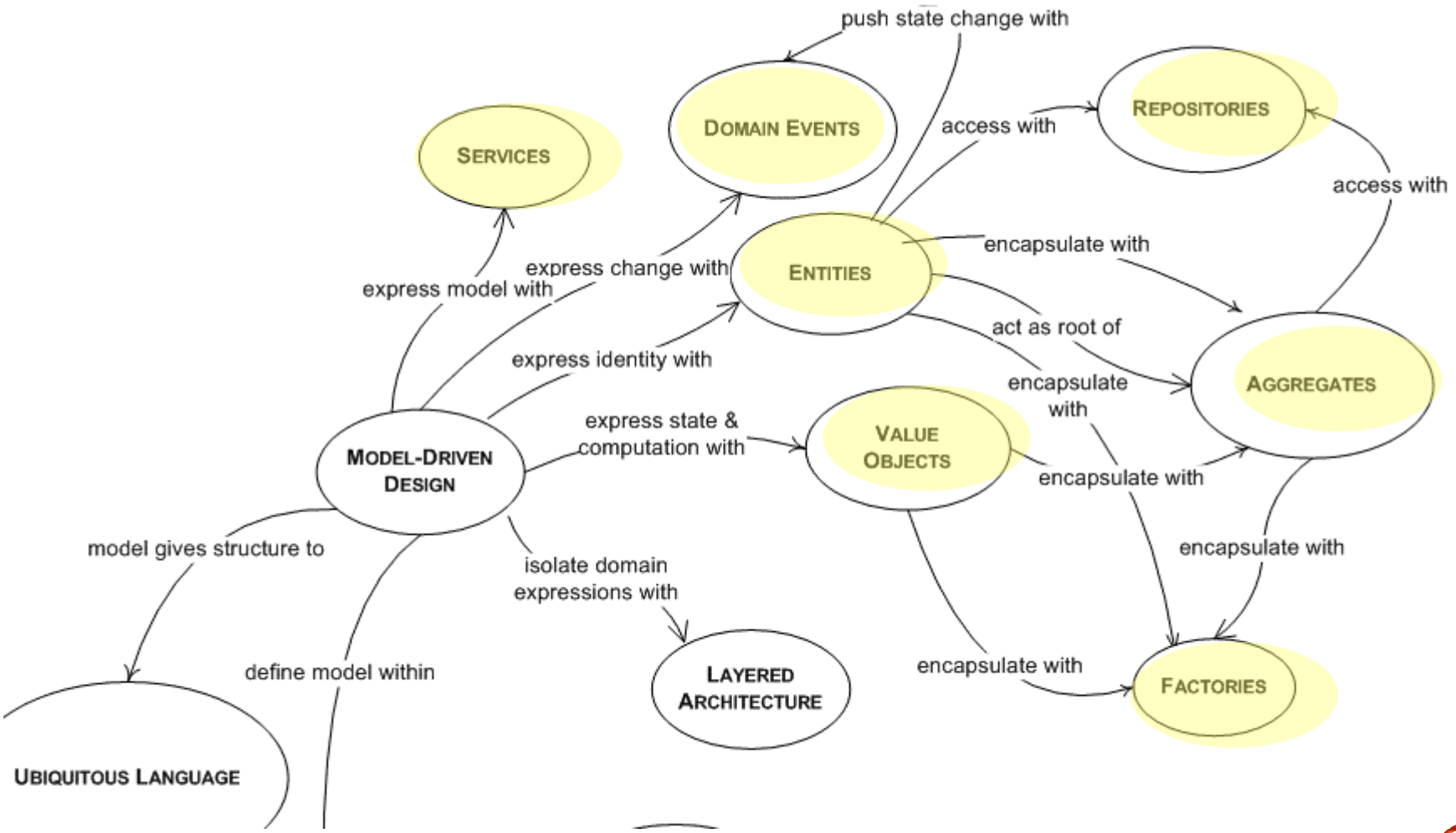




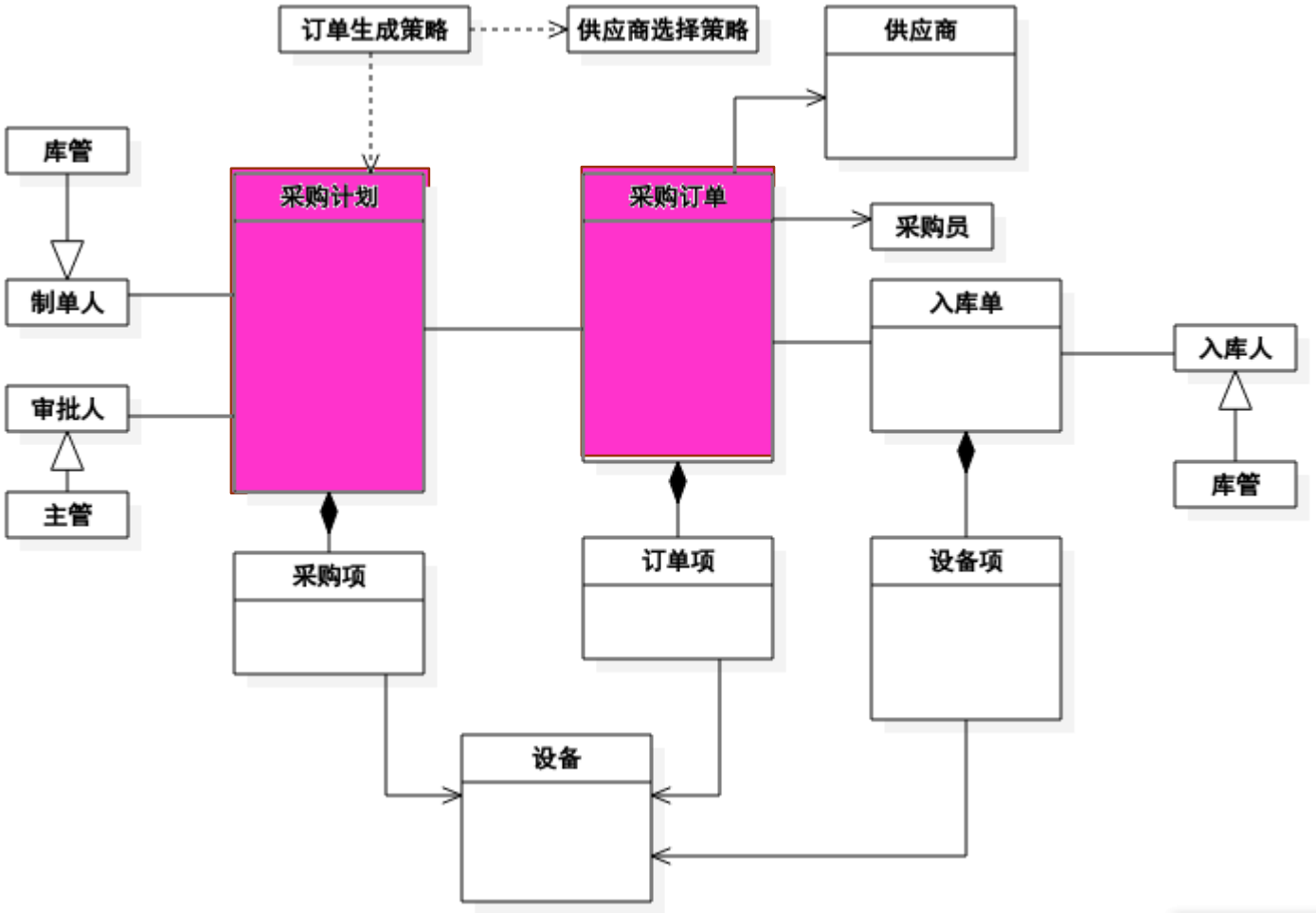
5

统一语言和实现

用DDD战术模式来实现领域模型和代码的高度统一



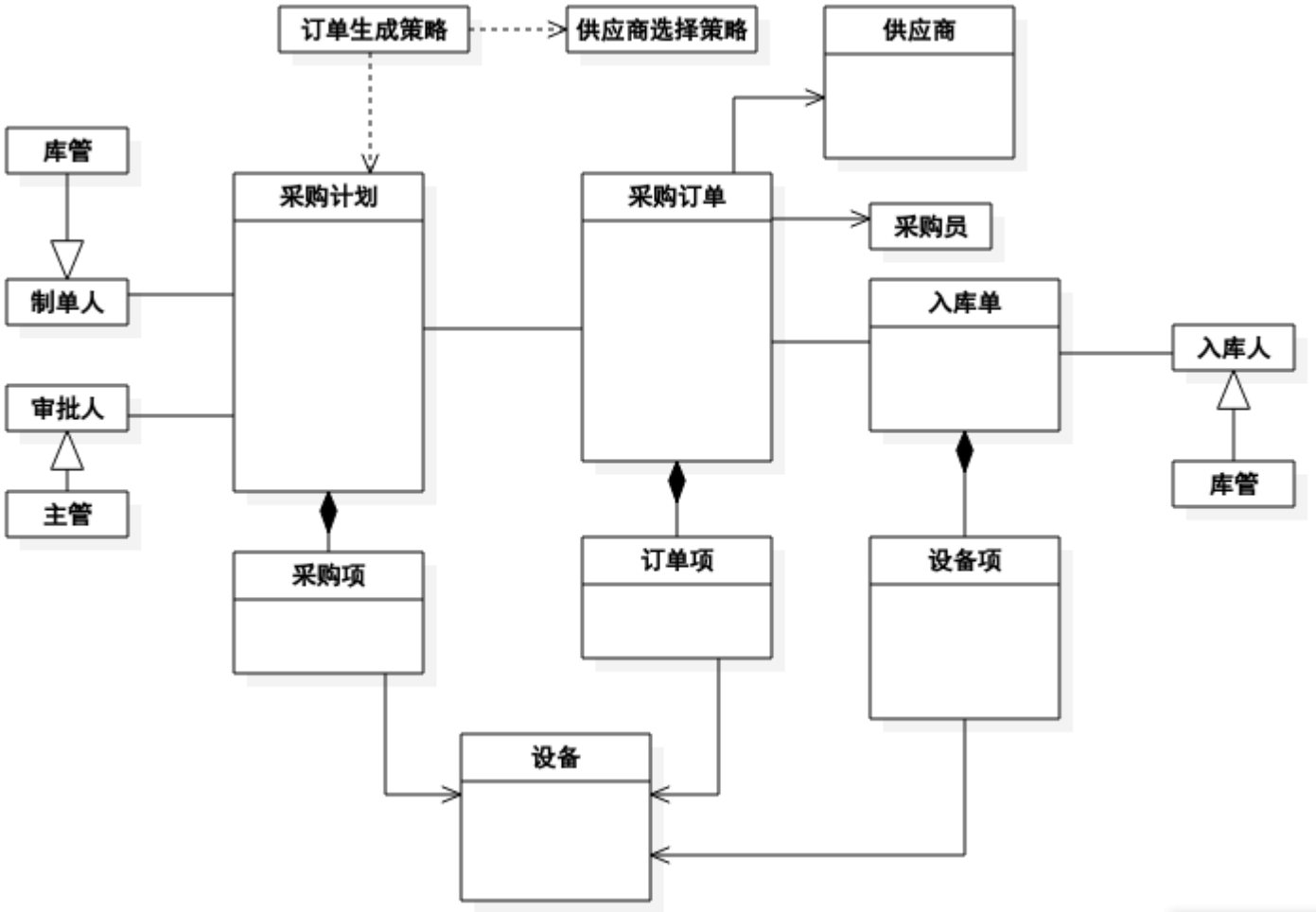
时标对象（Moment-Interval）比其他对象更值得关注



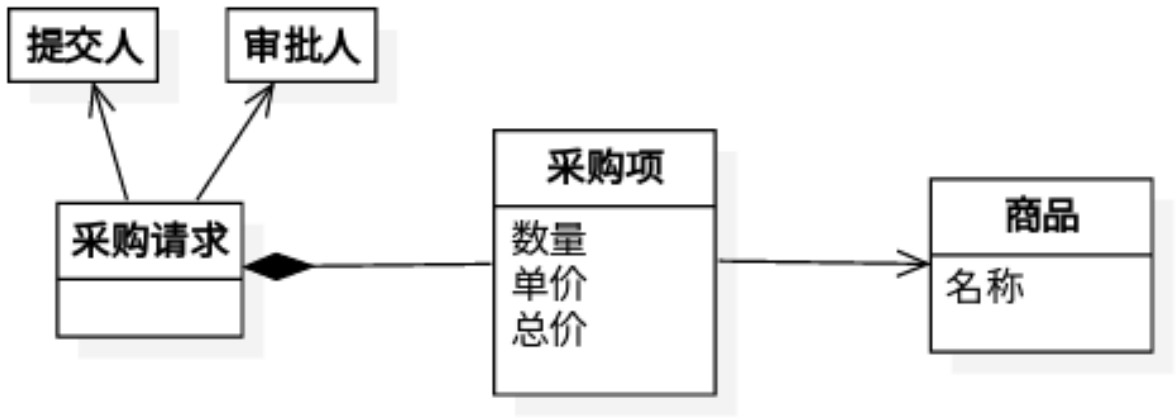
实体-和业务紧密相关

- 有状态
- 基于**标识**唯一识别一个对象
- 实体状态（属性）的变化不会影响到对象的识别

实体对象-值对象-领域事件-领域服务

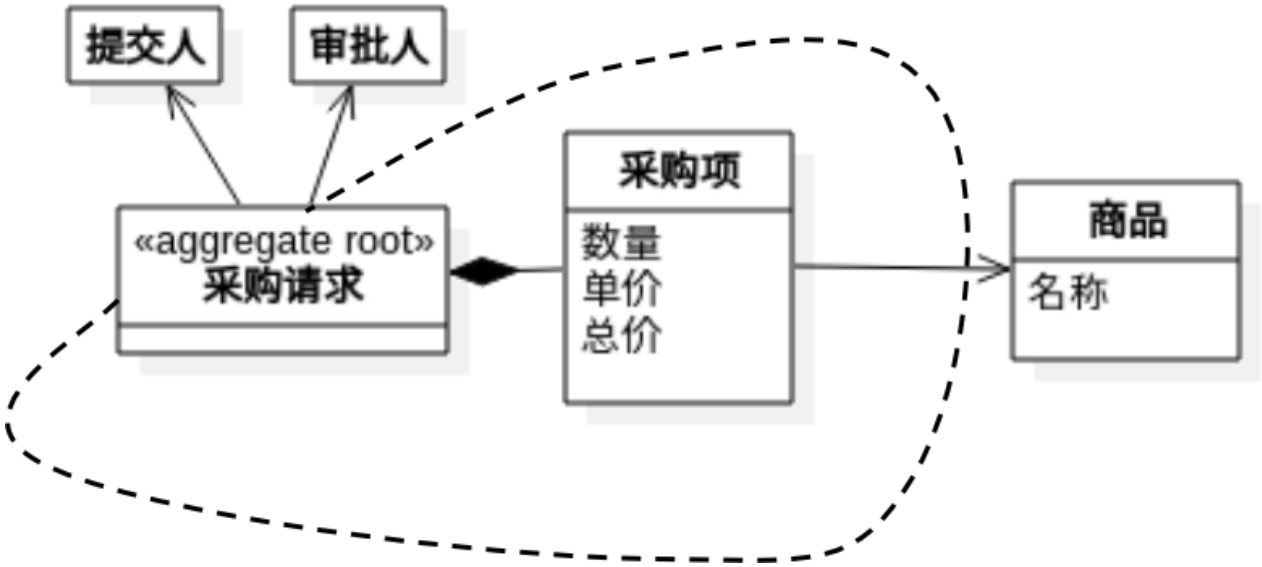


业务完整性问题和聚合




```
...
PurchaseRequest purchaseRequest = getPurchaseRequest(requestId);
PurchaseItem item = purchaseRequest.getItem(itemId);
item.setQuantity(1000);
savePurchaseItem(item);
```

业务完整性问题和聚合

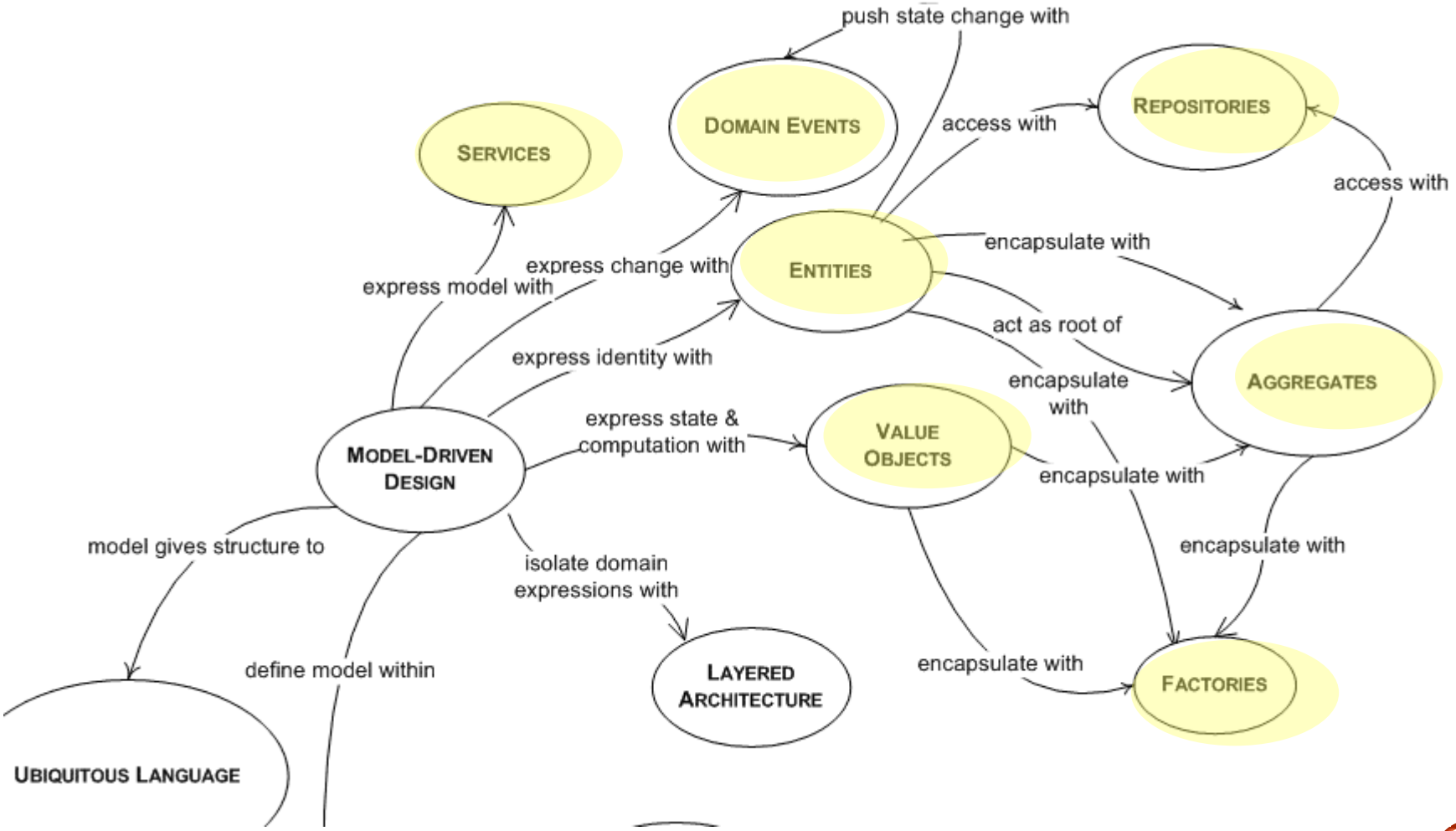


聚合

- 聚合是业务逻辑的边界。突破逻辑边界对内部对象进行操作，会破坏业务逻辑的完整性。

- 解决方案：所有对聚合的更新都应该通过聚合根进行。
  - 外部对象不允许直接访问聚合内的内部实体（可以使用，但不能持有）
  - 在聚合边界内，对象之间可以相互引用。但是，根是聚合中唯一允许被外部引用的元素，

小结



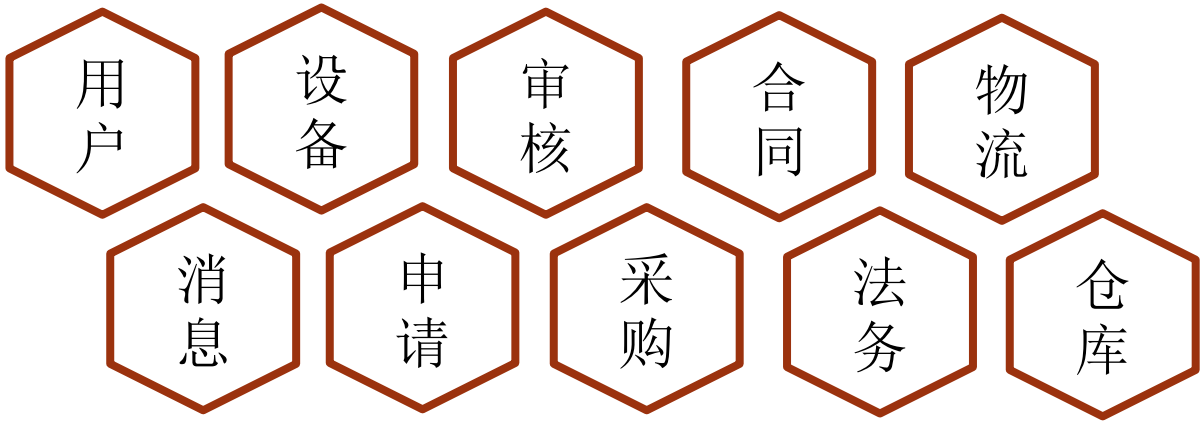
6

做一件事，把它做好

在业务域分解以简化复杂性和提升适应能力

有机的软件系统和无机的软件系统

有机的软件系统和无机的软件系统

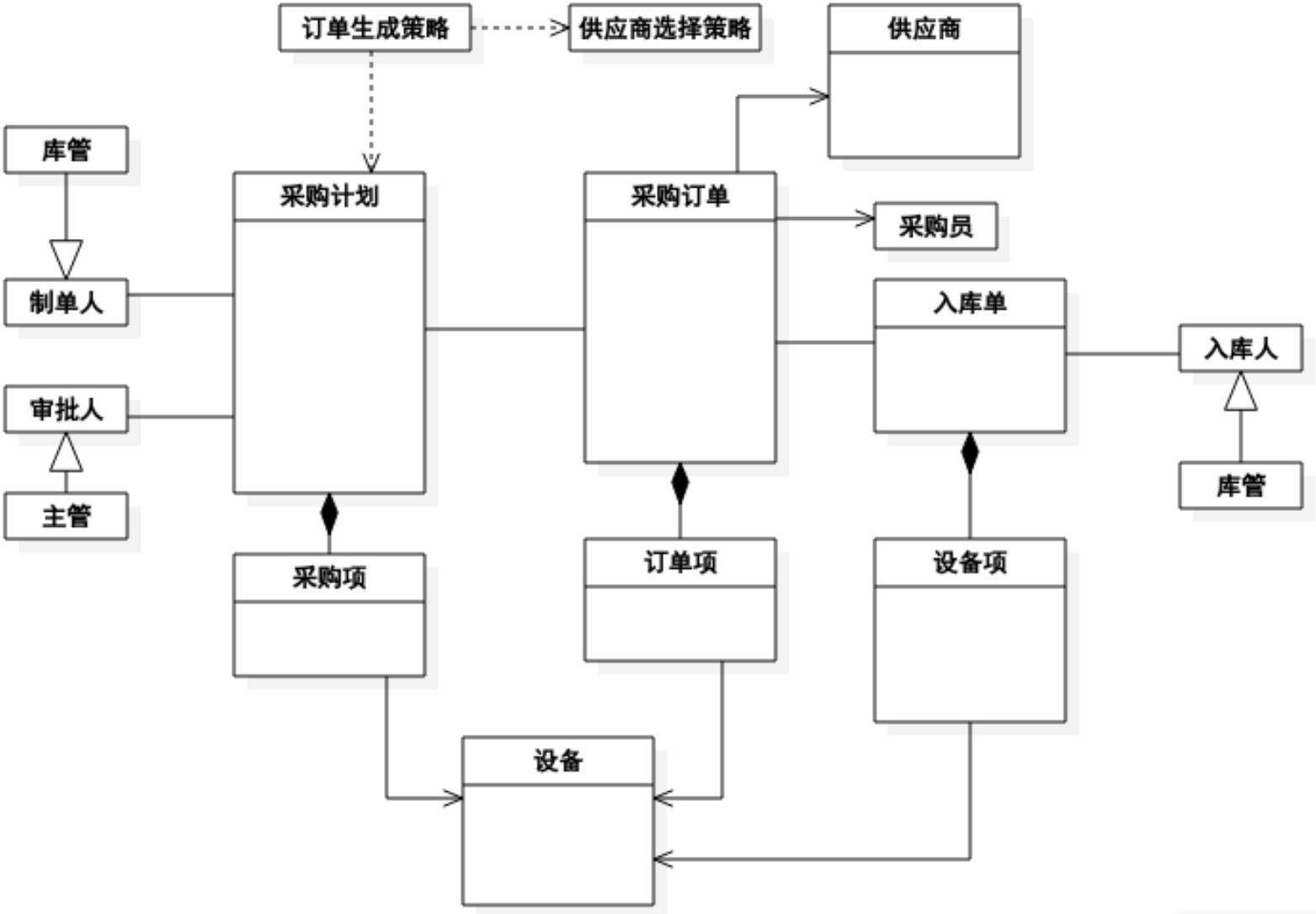


解耦没有本质关系的业务

```java
@Component
public class ReviewServiceImpl {
    @Autowired
    TextReviewRepository textReviewRepo;
    @Autowired
    LikeRepository likeReviewRepo;
    @Autowired
    EventPublisher eventPublisher;
    @Autowired
    private MemberService memberService;
    @Autowired
    ReviewInfoDTOHelper reviewInfoDTOHelper;
    @Autowired
    ReviewPointCounterService pointCounterService;
    @Autowired
    ReviewTimesCounterService timesCounterService;
    @Autowired
    CacheManager cacheManager;

    public ReviewServiceImpl() {
    }
```

```java
public class TextReview extends Review{
    private String comment;
    private boolean adminMessage = false;
    private String l1CommentId;
    private String rootResourceRefId;

    private String resourceName;
    private String resourceOwnerName;
    private String reviewedResourceAuthorId;

    private String tag;
    private String facetStr;
    private Float point;
    private String images;
```

解耦没有本质关系的业务



领域层独立的价值

- 能力通过领域层得到沉淀
- 数据通过领域层得到沉淀
- 商业竞争能力得到提升
- 业务响应能力和业务相应速度得到提升
- 服务质量得到提升

领域工程的发展

- 领域驱动设计相对于RUP的突破
  - 领域模型的作用被强调
  - 协作和演进的思维模式
  - 统一语言降低了表示差距
- 领域驱动设计相对于SPLE（软件产品线工程）的突破
  - 突出了子领域的概念
  - 是演进而非规划的观点

领域工程的发展

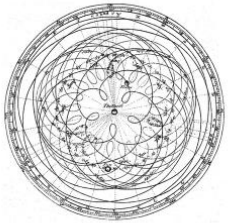
- 微服务对领域工程的推动
  - 带来了真正的边界
- 领域工程仍然解决的不好的地方
  - 统一语言的架构保证
  - 应用工程和领域工程的边界

领域模型

猜想与反驳

统一语言

1 模型的本质



4 事件风暴和领域建模

用事件风暴来发现和精化业务场景

事件风暴

2 从认知角度看领域建模

5 统一语言和实现

用DDD战术模式来实现领域模型和代码的高度统一

构造块，生命周期

3 统一语言和协作

业务、开发、测试紧密配合，
在澄清需求的过程中逐步完善领域模型

非技术因素

SBE/用例驱动

6 做一件事，把它做好

在业务域分解以简化复杂性和提升适应能力

VUCA时代

子域和BC

总结

- 领域工程是VUCA时代提升组织竞争力的一个有价值的工程方法
- 领域驱动设计的核心是领域模型和统一语言
- 猜想与反驳是领域模型逐步求精的方法
- 领域驱动设计的建模过程可以良好支持需求、实现和测试工作的实施

感谢参与

祝大家每天构建可持续生长的软件