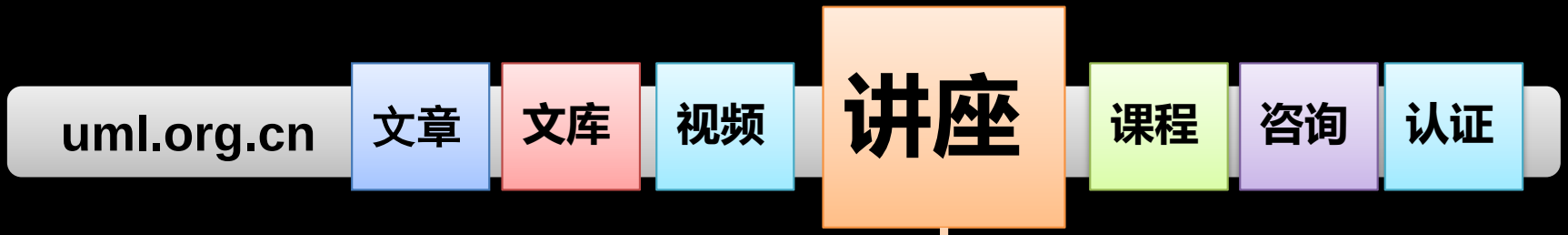
UML到底有什么用？

俎涛



火龙果讲堂

- 一线专家
- 案例回顾
- 经验分享

个人简介

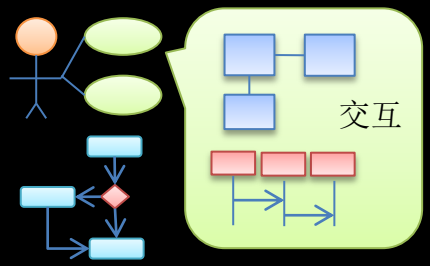
- 2001年创立了uml.org.cn，担任架构师。
- 2004年创立了IBM Rational用户组，
- 2005年获得 IBM Rational UG最佳主讲。
- 承担过20多个咨询项目。培训过300多家，例如：爱立信研发中心，诺基亚西门子研发中心，金蝶国际，阿里巴巴网络，腾讯科技、亚信科技、建行研发中心

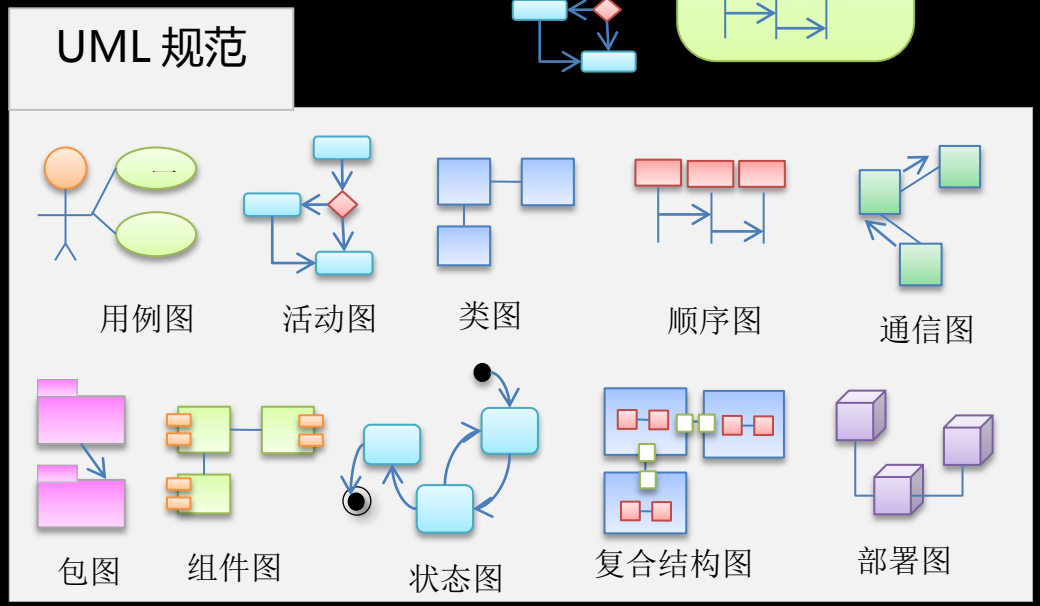
安排

- **什么是UML**
- UML有什么用处
- **UML的纠结**
- UML的失败故事
- UML的成功故事
- 建议：什么情况需要用UML，什么时候不用
- UML怎样才能学好

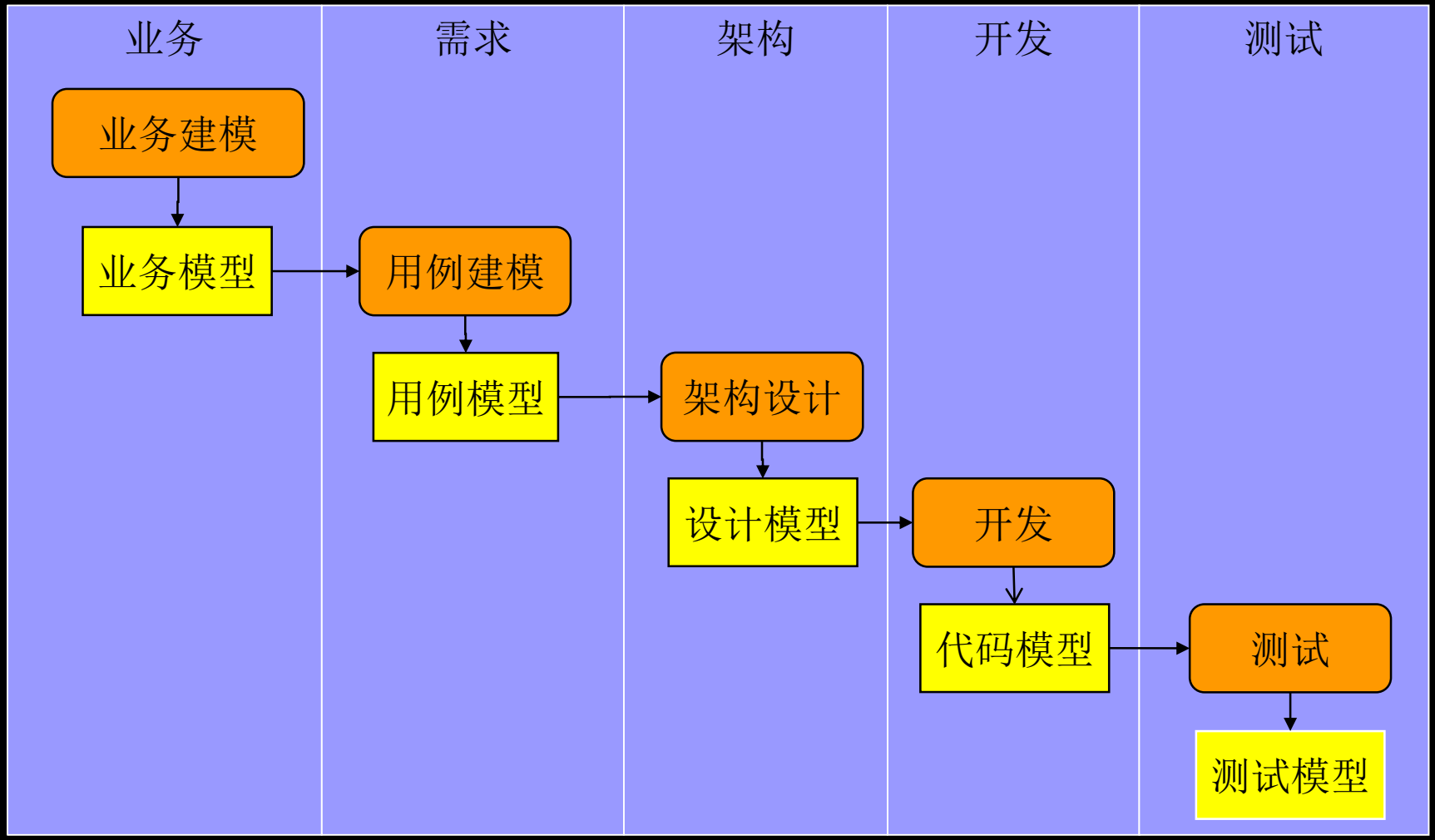
什么是UML

- UML是OMG 发布的建模规范之一
- 是目前最流行的软件建模语言
- 借鉴了软件行业很多传统建模





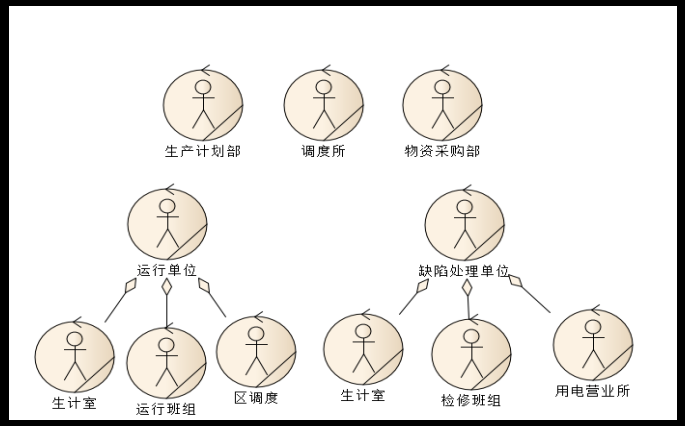
模型驱动的开发 路线图

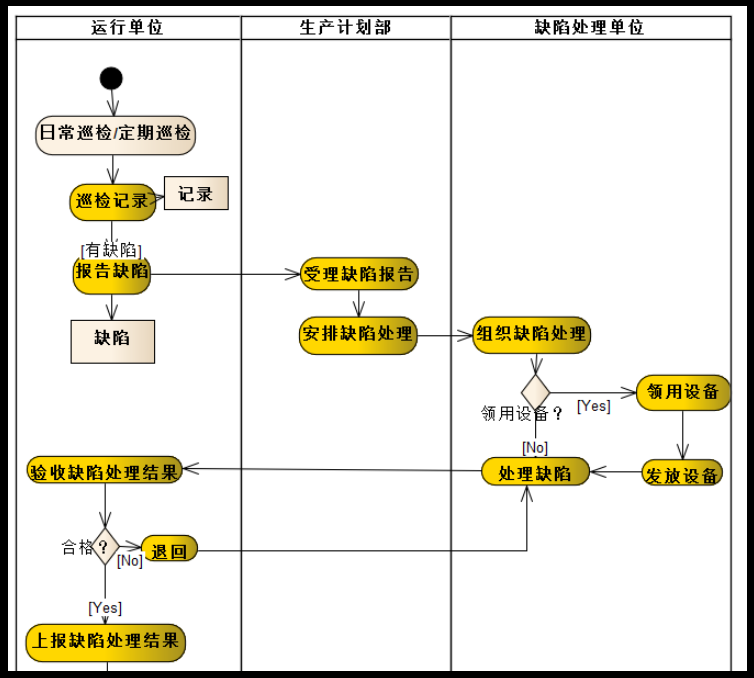


对不同角色的用处

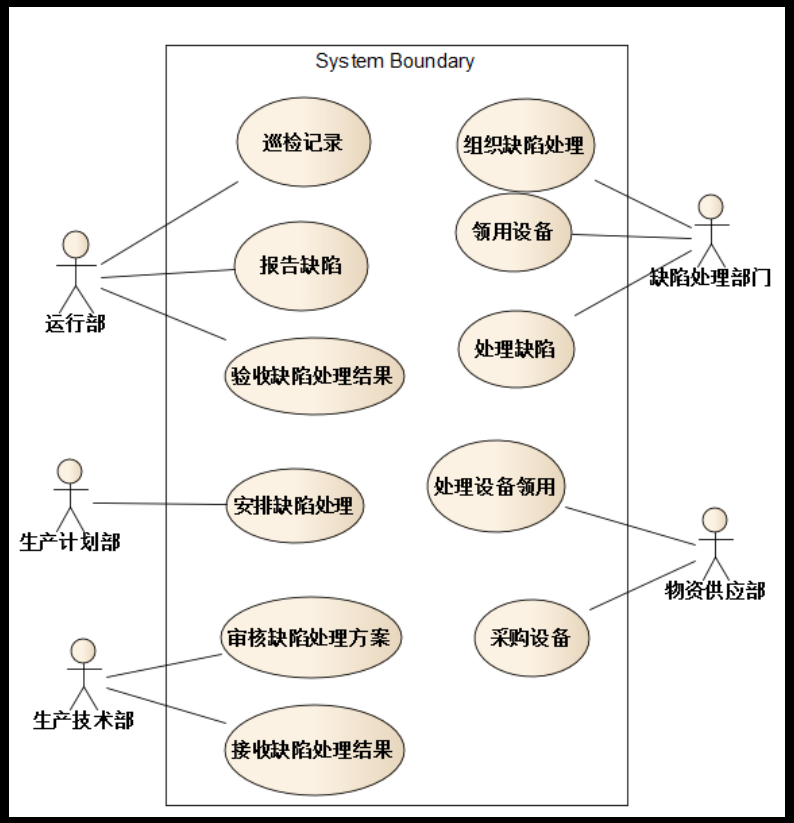
- 需求人员：业务建模，系统建模
- 架构人员：架构建模
- 数据库设计人员：数据建模
- 开发人员：详细设计+代码建模
- 测试人员：测试建模

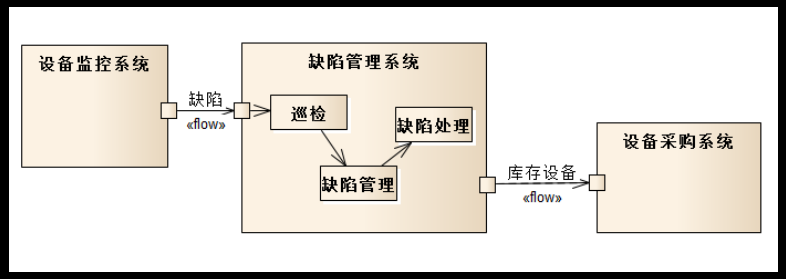
业务模型


生产计划部
调度所
物资采购部
运行单位
缺陷处理单位
生计室
运行班组
区调度
生计室
检修班组
用电营业所

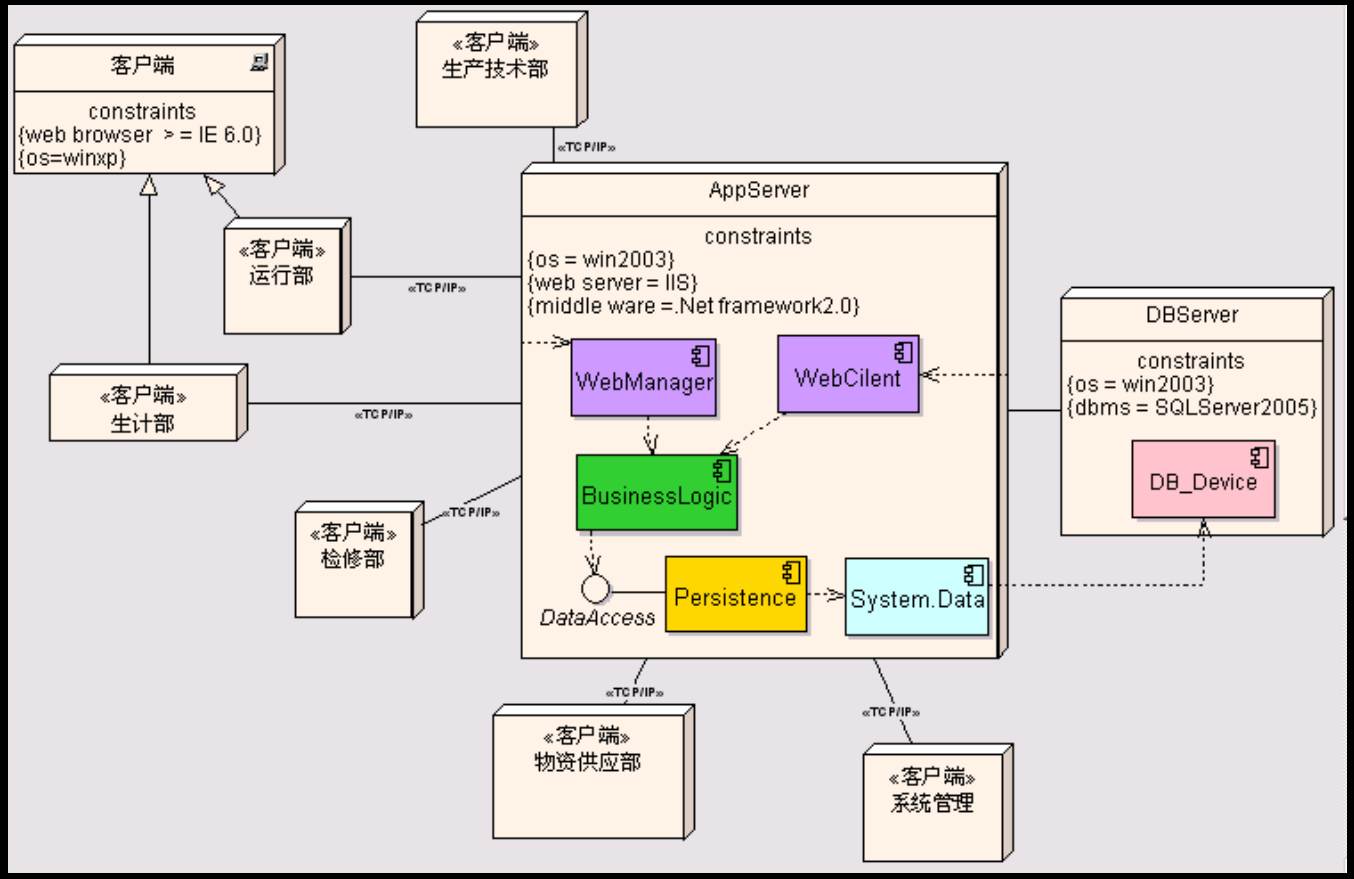

运行单位
生产计划部
缺陷处理单位
日常巡检/定期巡检
巡检记录
记录
[有缺陷]
报告缺陷
缺陷
受理缺陷报告
安排缺陷处理
组织缺陷处理
领用设备？
[Yes]
领用设备
[No]
处理缺陷
发放设备
验收缺陷处理结果
合格？
[No]
退回
[Yes]
上报缺陷处理结果

需求建模

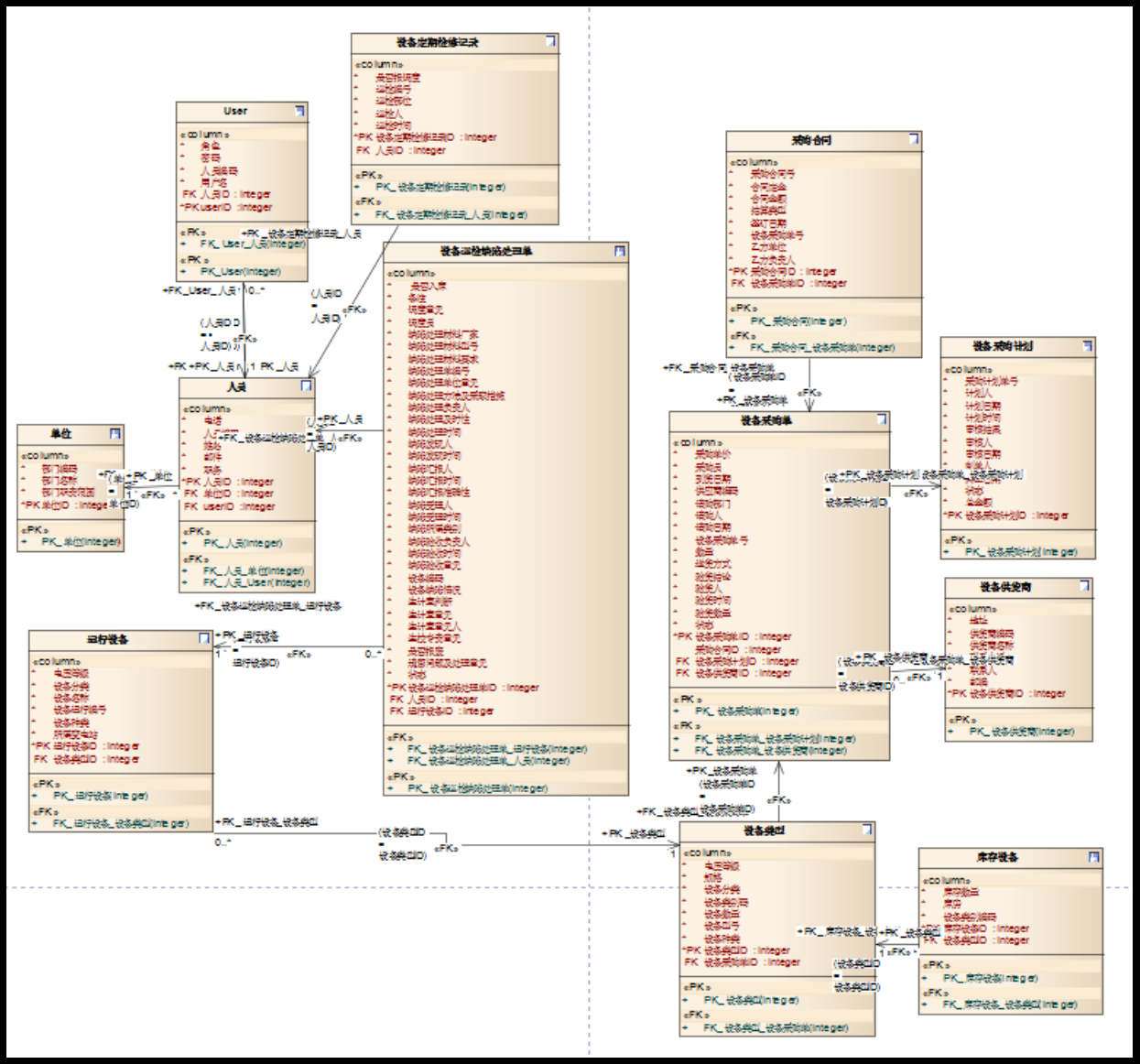

System Boundary
巡检记录
报告缺陷
验收缺陷处理结果
组织缺陷处理
领用设备
处理缺陷
处理设备领用
采购设备
安排缺陷处理
审核缺陷处理方案
接收缺陷处理结果
运行部
生产计划部
生产技术部
缺陷处理部门
物资供应部


设备监控系统
缺陷
«flow»
缺陷管理系统
巡检
缺陷处理
缺陷管理
库存设备
«flow»
设备采购系统

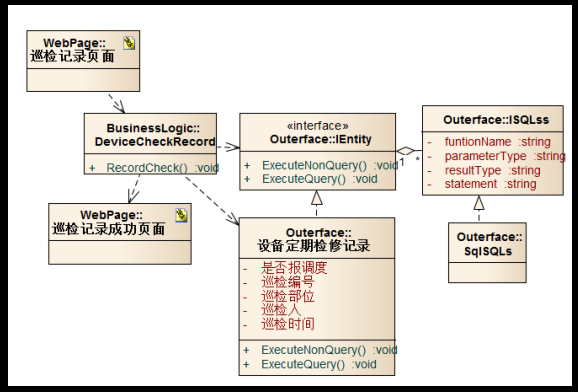
架构模型

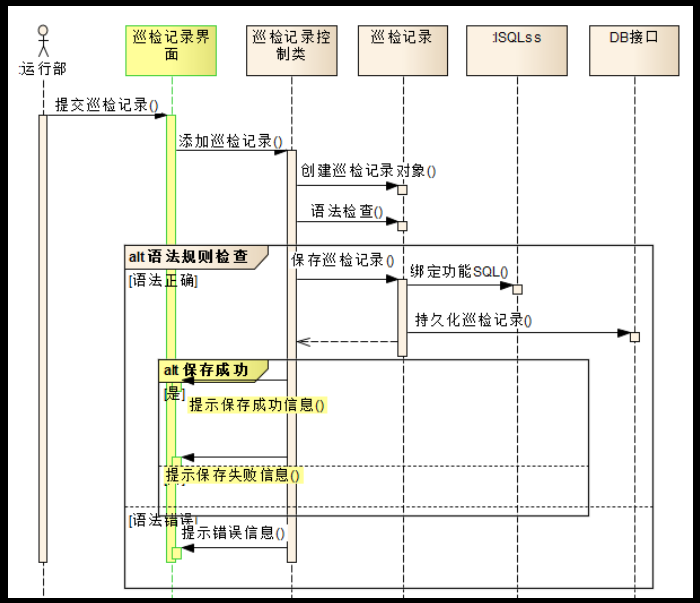

客户端
constraints
{web browser > = IE 6.0}
{os=winxp}
«客户端»
生产技术部
«TCP/IP»
AppServer
constraints
{os = win2003}
{web server = IIS}
{middle ware =.Net framework2.0}
«客户端»
运行部
«TCP/IP»
«客户端»
生计部
«TCP/IP»
«客户端»
检修部
«TCP/IP»
WebManager
WebCilent
BusinessLogic
DataAccess
Persistence
System.Data
DBServer
constraints
{os = win2003}
{dbms = SQLServer2005}
DB_Device
«TCP/IP»
«客户端»
物资供应部
«TCP/IP»
«客户端»
系统管理

数据模型


设备定期检修记录
User
人员
单位
设备运检纳统处理单
运行设备
采购合同
设备采购计划
设备采购单
设备供货商
设备类型
库存设备

代码建模

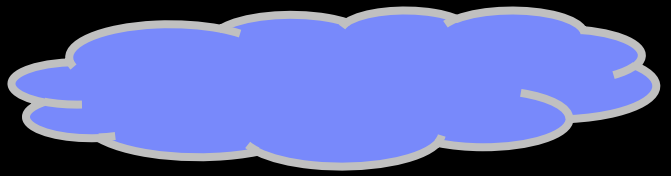

WebPage::
巡检记录页面
BusinessLogic::
DeviceCheckRecord
+ RecordCheck() :void
«interface»
Outerface::IEntity
+ ExecuteNonQuery() :void
+ ExecuteQuery() :void
Outerface::ISQLss
- funtionName :string
- parameterType :string
- resultType :string
- statement :string
WebPage::
巡检记录成功页面
Outerface::
设备定期检修记录
- 是否报调度
- 巡检编号
- 巡检部位
- 巡检人
- 巡检时间
+ ExecuteNonQuery() :void
+ ExecuteQuery() :void
Outerface::
SqlSQLs


运行部
巡检记录界面
巡检记录控制类
巡检记录
ISQLss
DB接口
提交巡检记录()
添加巡检记录()
创建巡检记录对象()
语法检查()
alt 语法规则检查
[语法正确]
保存巡检记录()
绑定功能SQL()
持久化巡检记录()
alt 保存成功
[是]
提示保存成功信息()
提示保存失败信息()
[语法错误]
提示错误信息()

UML的纠结

- 辉煌
  - 被认为是有史以来最成功、应用最广的IT行业规范
  - 其他的建模要不消亡，要不成为UML的子集
  - 越来越多的职位期望掌握UML，

- 失落
  - 学习的人很多，真正能用好的人少
  - UML不过是一组符号集
  - UML被当作形式主义



UML的失败故事

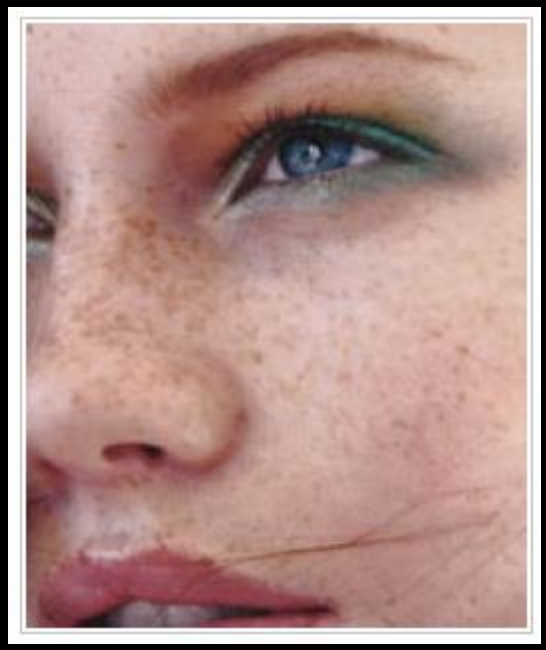
故事1：某航空公司

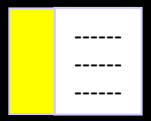
- 用UML 的失败了
- 没有用UML的成功了

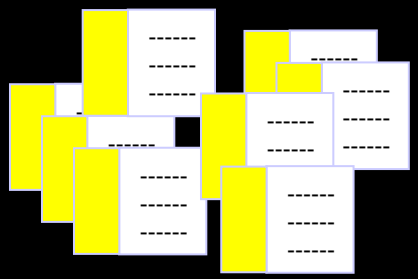
讨论一下：为什么失败

失败教训

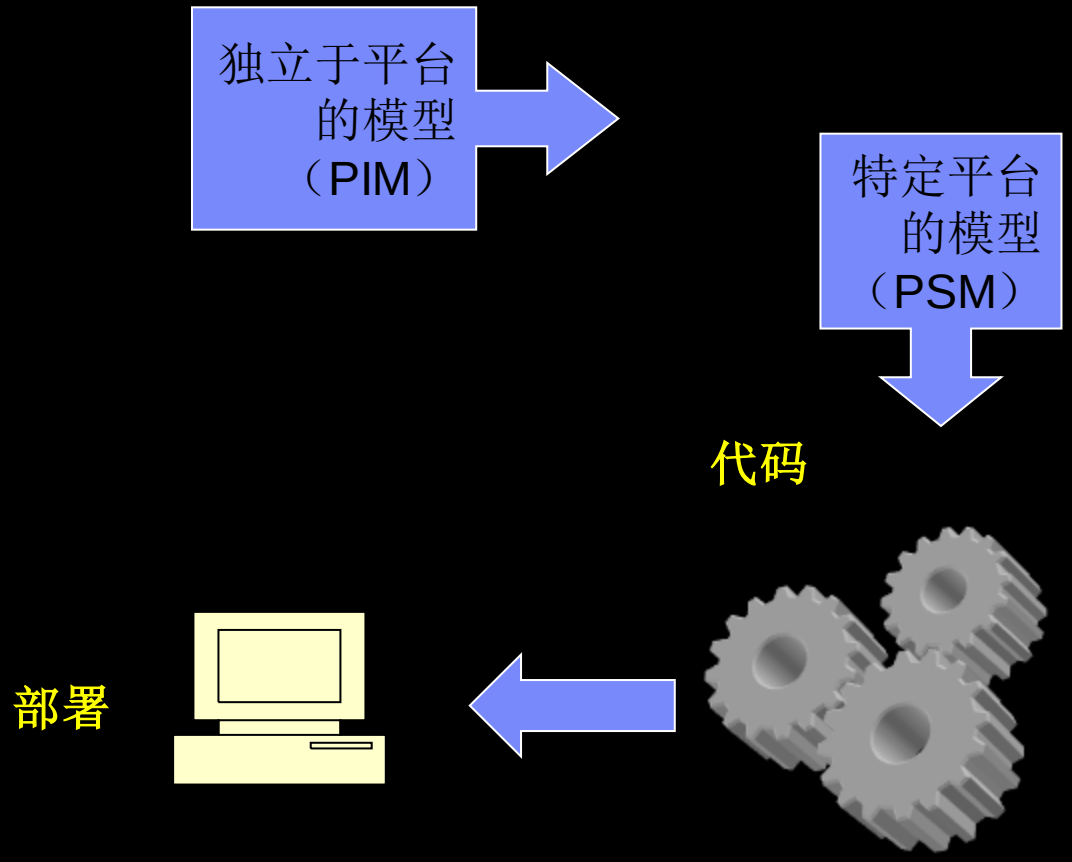
- 一个很美，100个很丑







故事2：某MDA 项目



讨论一下：为什么失败

失败教训

- 把 UML当作了开发语言
- 因为 UML过度依赖第三方框架
- 把所有开发人员都当作了设计师

UML的成功故事

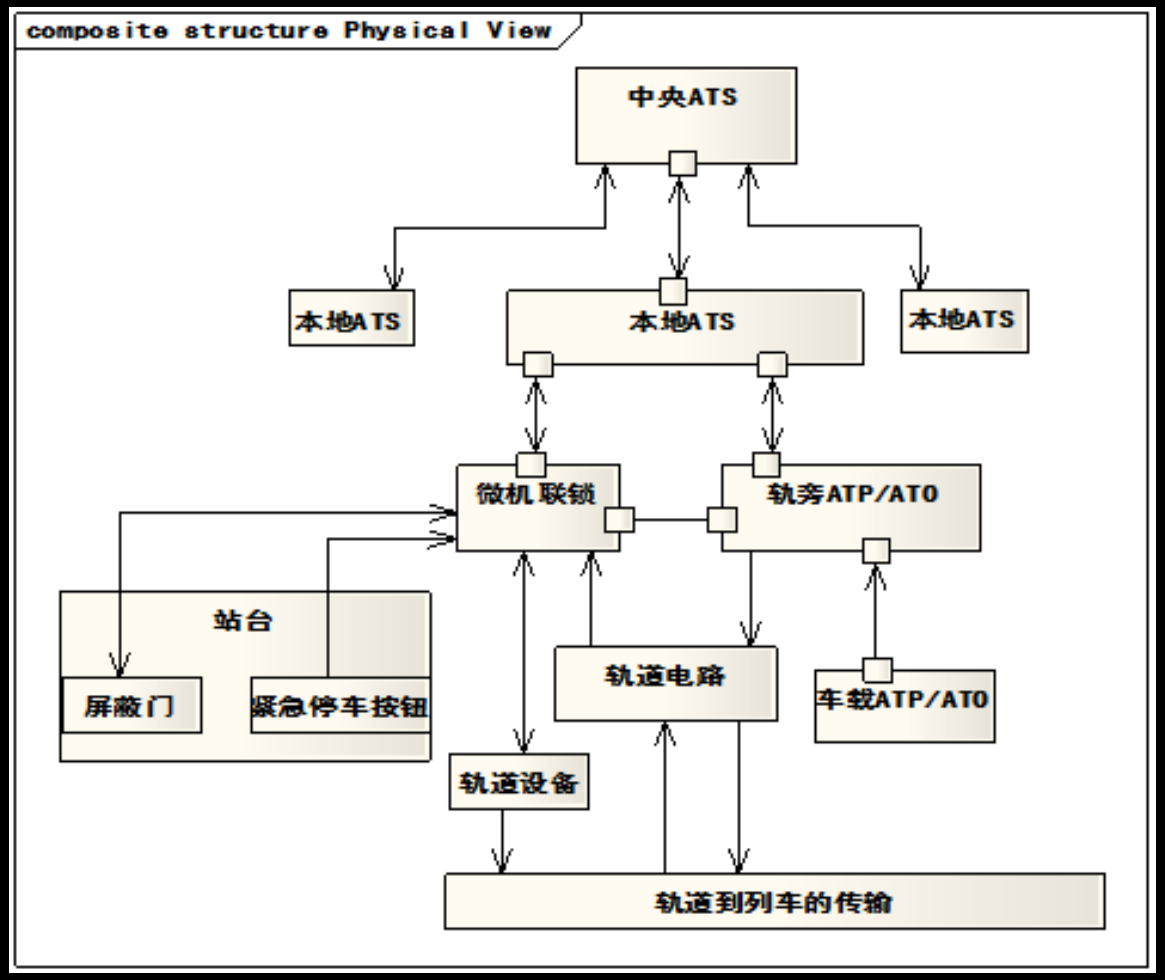
故事 1 ：某轨道交通监控系统

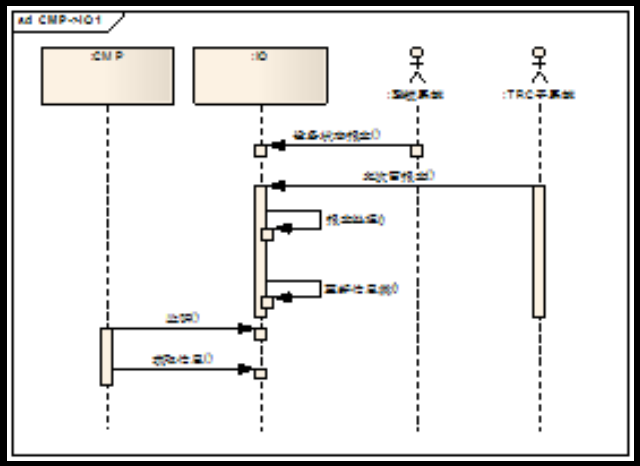
- 回放
  - 分布式、实时、多任务系统
  - 新领域项目
  - 开发 1 年多，集成 + 仿真测试时候出了问题
    - 难以集成
    - 可靠性问题、性能问题
    - 总也改不好

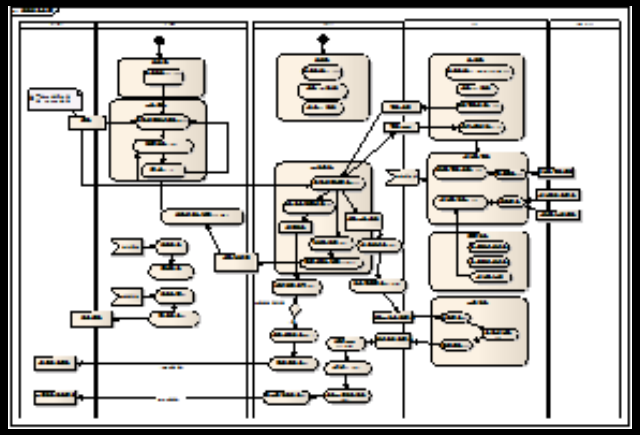
UML建模的应用

- 对已有的需求和设计进行逆向建模
- 诊断问题、定位问题
- 重构需求和设计
- 修改已有系统
- 测试与验证

图例


composite structure Physical View
中央ATS
本地ATS
本地ATS
本地ATS
微机联锁
轨旁ATP/ATO
站台
屏蔽门
紧急停车按钮
轨道电路
车载ATP/ATO
轨道设备
轨道到列车的传输





讨论一下：为什么成功

成功经验

- 引入UML后明确了需求和设计的内容
- 模型清晰地描述了系统架构，对出现的问题能够准确定位，分析影响范围
- 模型提供了持续设计的基础：
  - 性能、可靠性、扩展和复用
- UML让系统变得简单，可以更专注于实现

应用者的困惑

- 讨论：
  - 开始的时候觉得UML可能有用
  - 中间觉得ＵＭＬ很有用
  - 后来觉得ＵＭＬ好像又没有什么用了

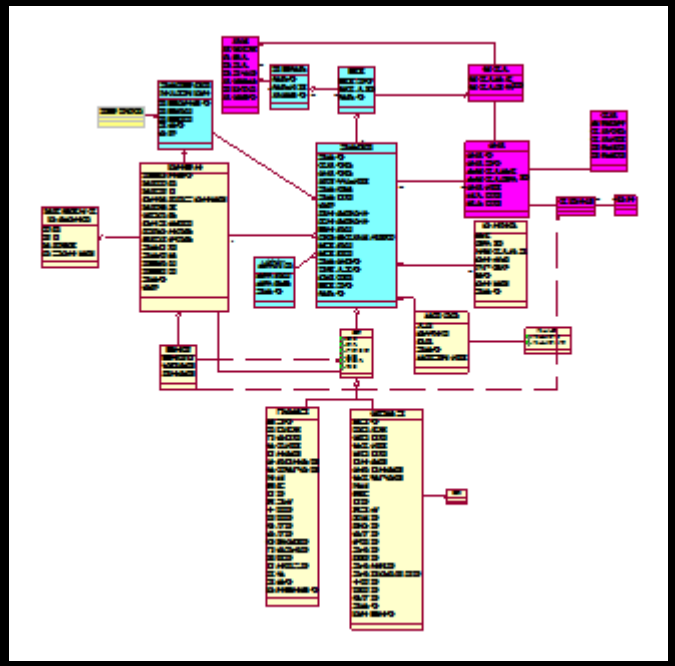
故事2：某保险公司业务系统

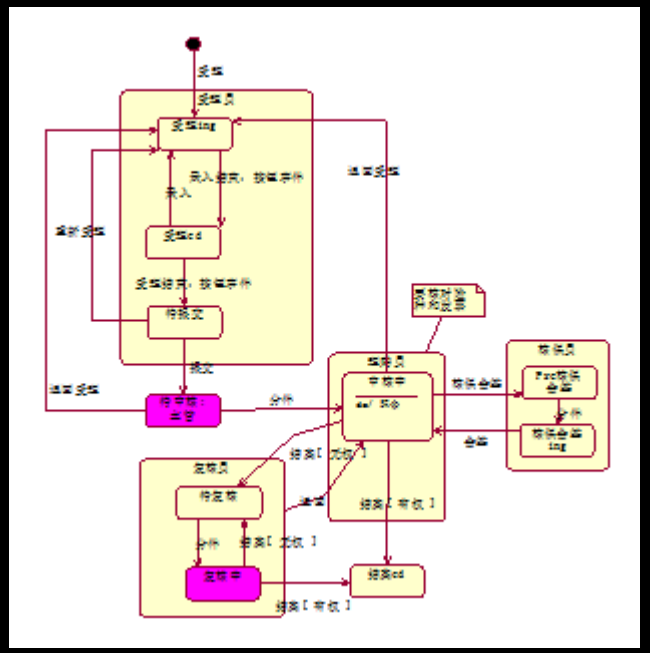
- 故事回放
  - 进行了多期项目
  - 看似不复杂的业务，需求却总在变化，总是漏掉点什么，造成变更成本
  - 不断地迭代，不断地修改，没有基点，也没有终点
  - 用户对最终的系统总是不满意，
  - 项目延期

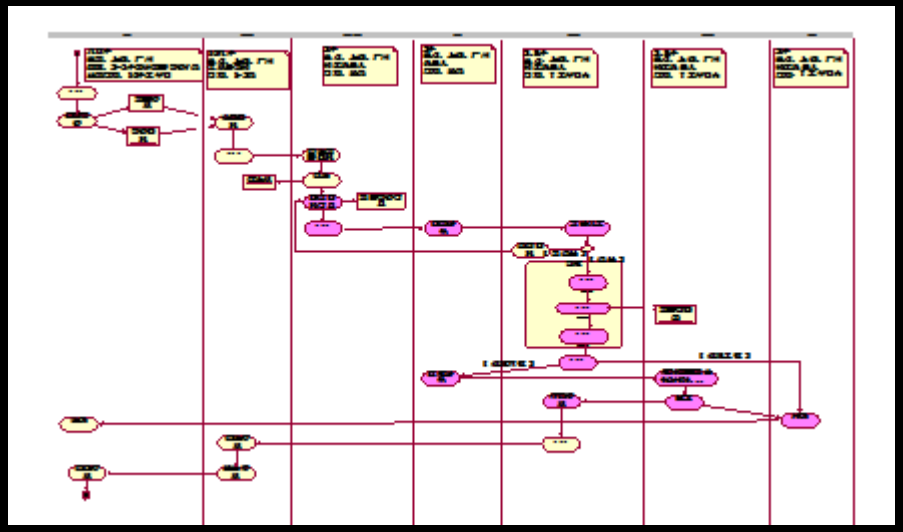
UML建模应用方法

- 建立模型驱动的开发过程
  - 业务模型
  - 需求模型
  - 架构模型
  - 详细设计模型
  - 测试模型
- 在理清需求和架构的基础上迭代开发
- 提高了面向用户的预见力，实现了有序迭代
- 提前 2 个月完成，用户满意！

图例







讨论一下：为什么成功

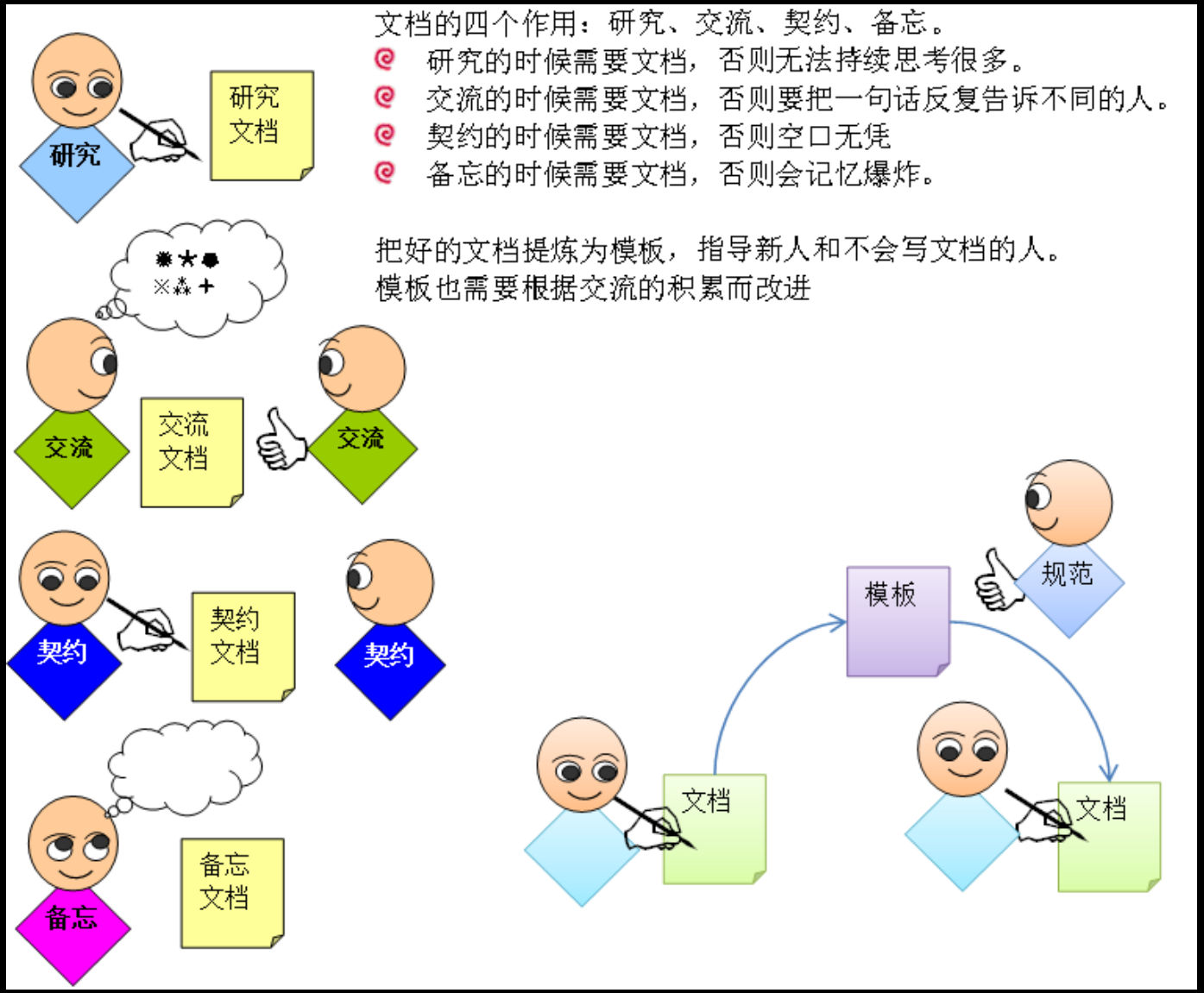
成功经验

- 业务：
  - 通过建立业务模型，建立了业务人员和IT的桥梁。
- 需求：
  - 不只停留在界面原型，还分析并建模了功能，接口、性能、扩展、安全需求。
- 设计
  - 明确了架构，并对系统多变部分充分进行了扩展设计，为开发提供了更具体的指导。
- 开发
  - 基于明确的需求和设计，更加高效而有序。
- 过程：
  - 让迭代有稳定的基点

UML有什么用处

- UML是文档的精髓
- 先看看文档有什么用处

文档-可持续的交流



文档的四个作用：研究、交流、契约、备忘。

- 研究的时候需要文档，否则无法持续思考很多。
- 交流的时候需要文档，否则要把一句话反复告诉不同的人。
- 契约的时候需要文档，否则空口无凭
- 备忘的时候需要文档，否则会记忆爆炸。

把好的文档提炼为模板，指导新人和不会写文档的人。
模板也需要根据交流的积累而改进

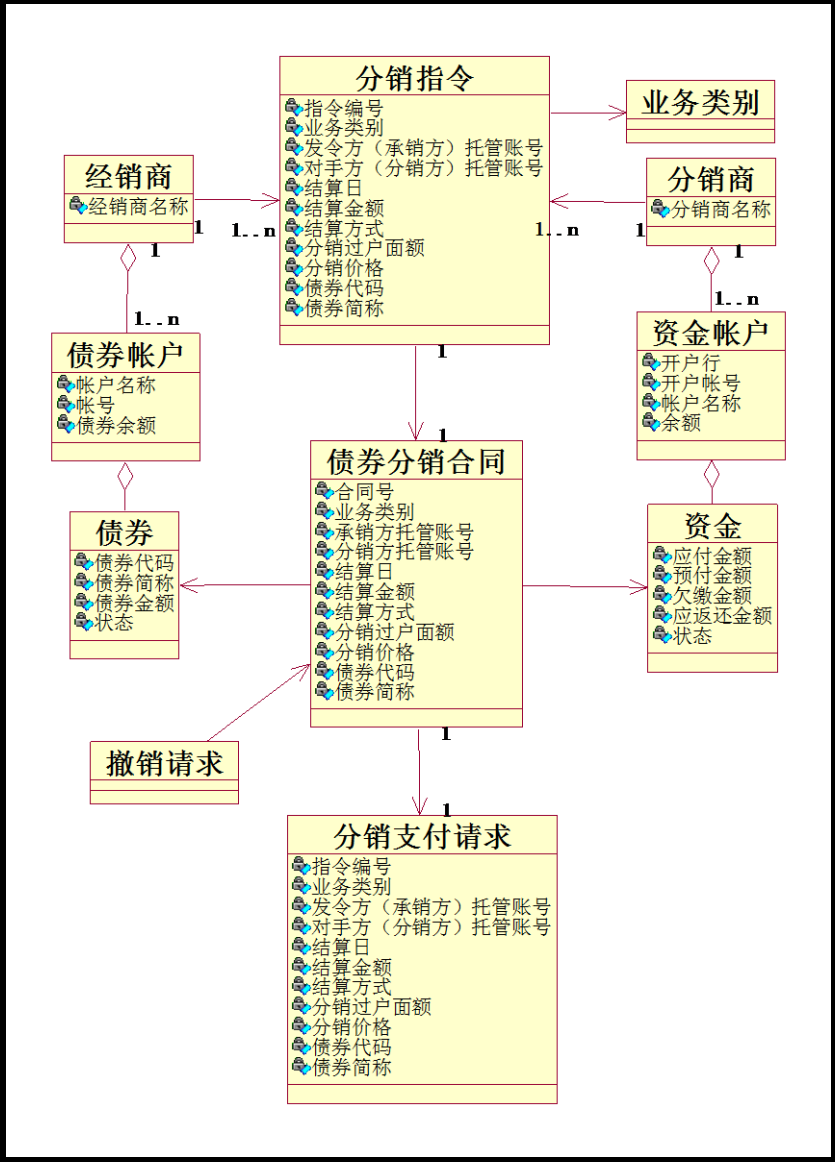
晦涩的文字怎么办

你读的懂么

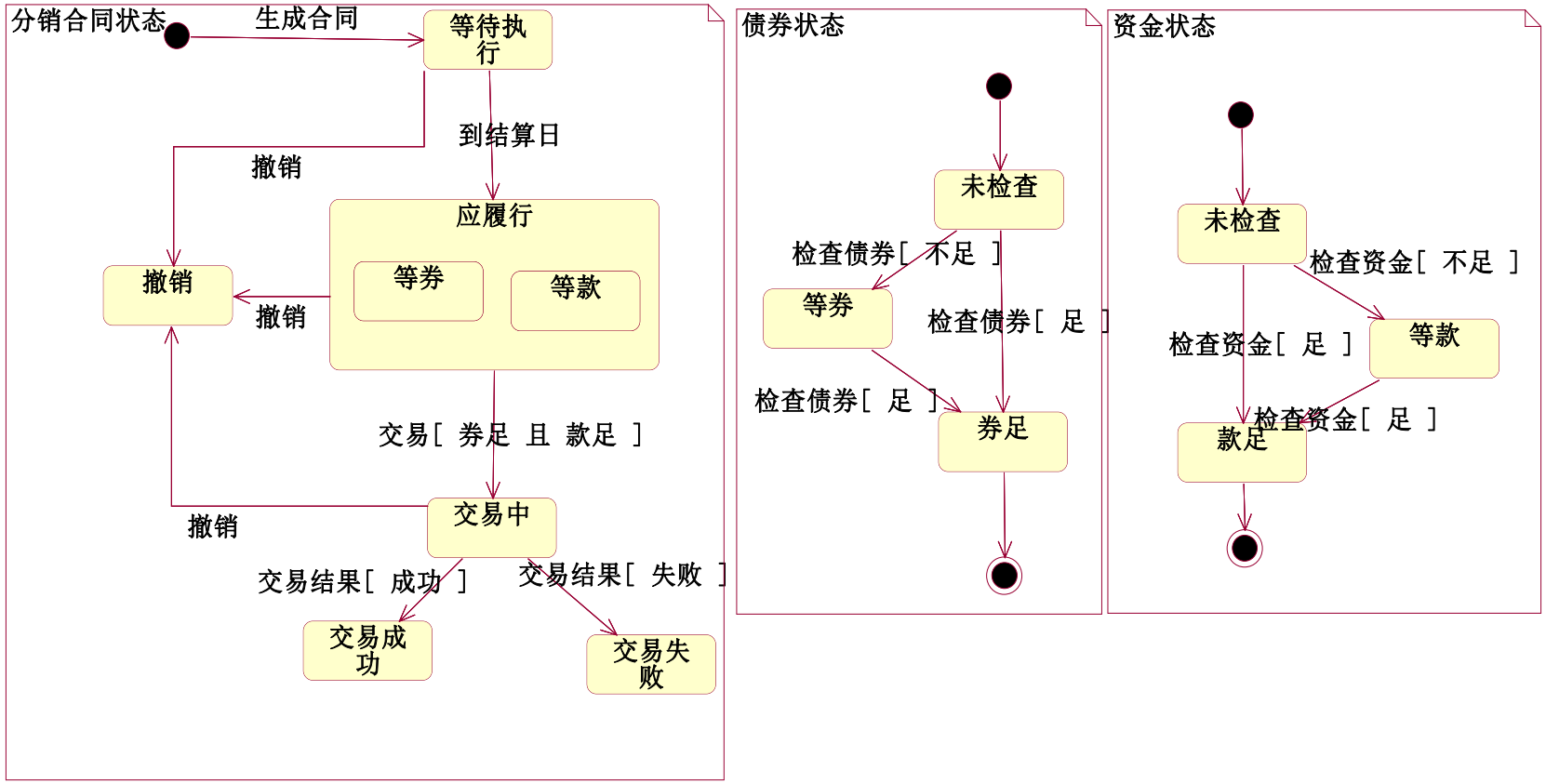
- 如果“是否 直接借记承销商 清算帐户 划付发行认购款”的取值为“Yes”，则按以下流程处理。
- 在结算日，合同初始状态为“应履行”、债券状态为“未检查”，资金状态为“未检查”，撤销状态为“未撤销”。若合同状态为“应履行”/“等券”、 债券状态为“未检查”/“等券”，资金状态为“未检查”，撤销状态为“未撤销”。簿记系统检查分销卖出方承销额度科目（0401）余额是否足额。如果不足额，检查其承销额度科目（0401）余额和待确认债权科目（0403）余额合计是否足额。如不足且合同债券状态为“未检查”，则将债券状态改为“等券”，合同等待下一次执行。

从文字到可视化建模

- 实体对象



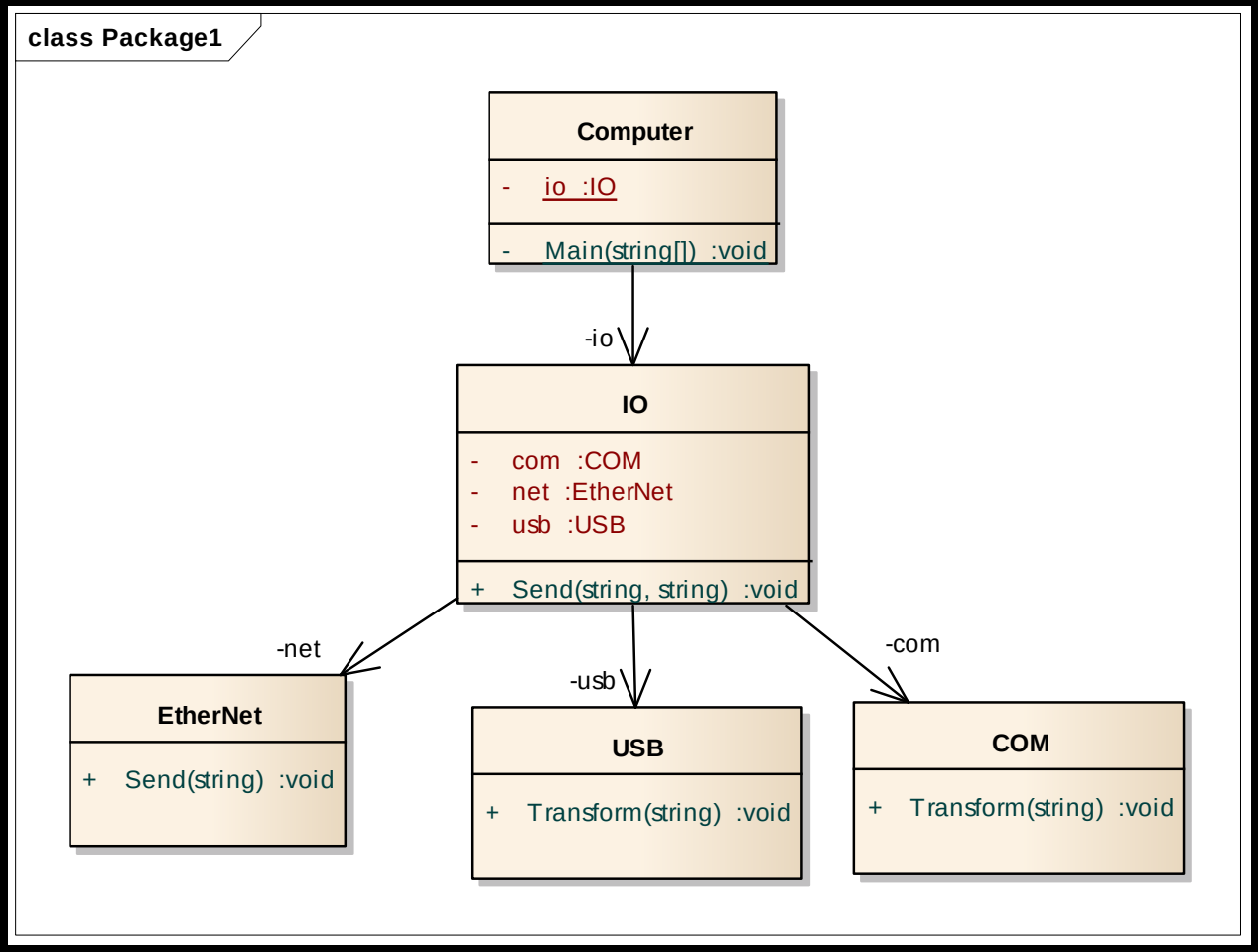
实体状态


分销合同状态
生成合同
等待执行
到结算日
撤销
应履行
撤销
等券
等款
撤销
交易[券足 且 款足]
撤销
交易中
交易结果[成功]
交易结果[失败]
交易成功
交易失败
债券状态
未检查
检查债券[不足]
等券
检查债券[足]
检查债券[足]
券足
资金状态
未检查
检查资金[不足]
检查资金[足]
等款
检查资金[足]
款足

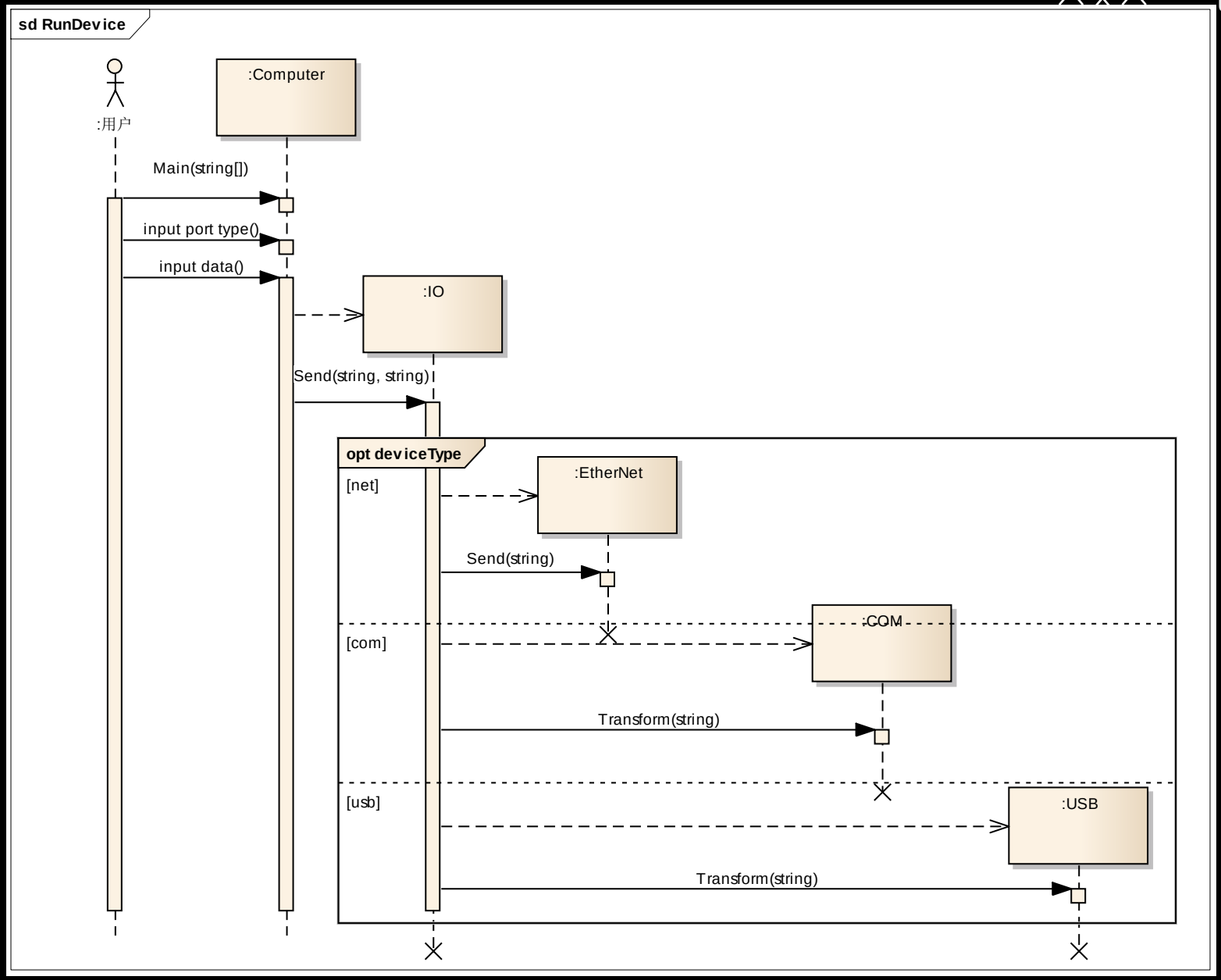
代码也是一种文档——工作脚本


```
namespace DiscoupleDevice
{
    class EtherNet
    {
        public void Send(string data)...
    }
    class COM...
    class USB...

    class IO
    {
         private EtherNet net;
         private COM com;
         private USB usb;

        public void Send(string data, string port)...
    }
    class Computer
    {
        static private IO io;
        [STAThread]
        static void Main(string[] args)
        {
            Console.WriteLine(" 请指定设备端口： ");
            string port = Console.ReadLine().ToLower();
            Console.WriteLine(" 请输入要发送的数据： ");
            string data = Console.ReadLine().ToLower();
            io = new IO();
            io.Send(data, port);
        }
    }
}
```

如何建模代码结构


class Package1
Computer
- io :IO
- Main(string[]) :void
-io
IO
- com :COM
- net :EtherNet
- usb :USB
+ Send(string, string) :void
-net
-usb
-com
EtherNet
+ Send(string) :void
USB
+ Transform(string) :void
COM
+ Transform(string) :void

建模代码行为-顺序图


sd RunDevice
:用户
:Computer
Main(string[])
input port type()
input data()
:IO
Send(string, string)
opt deviceType
[net]
:EtherNet
Send(string)
[com]
:COM
Transform(string)
[usb]
:USB
Transform(string)

什么情况需要用UML

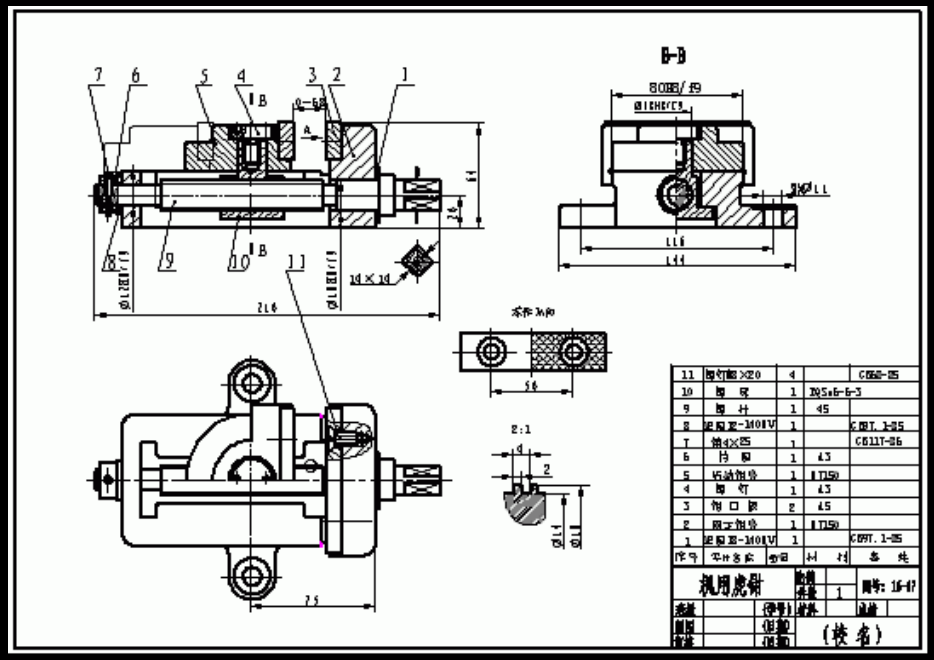
- 复杂项目
- 集成项目
- 期望在逻辑深度上超越竞争者的人
- 持续建设、运营的系统或产品建设
- 嵌入式系统
- 不断扩展的团队　+　大规模团队

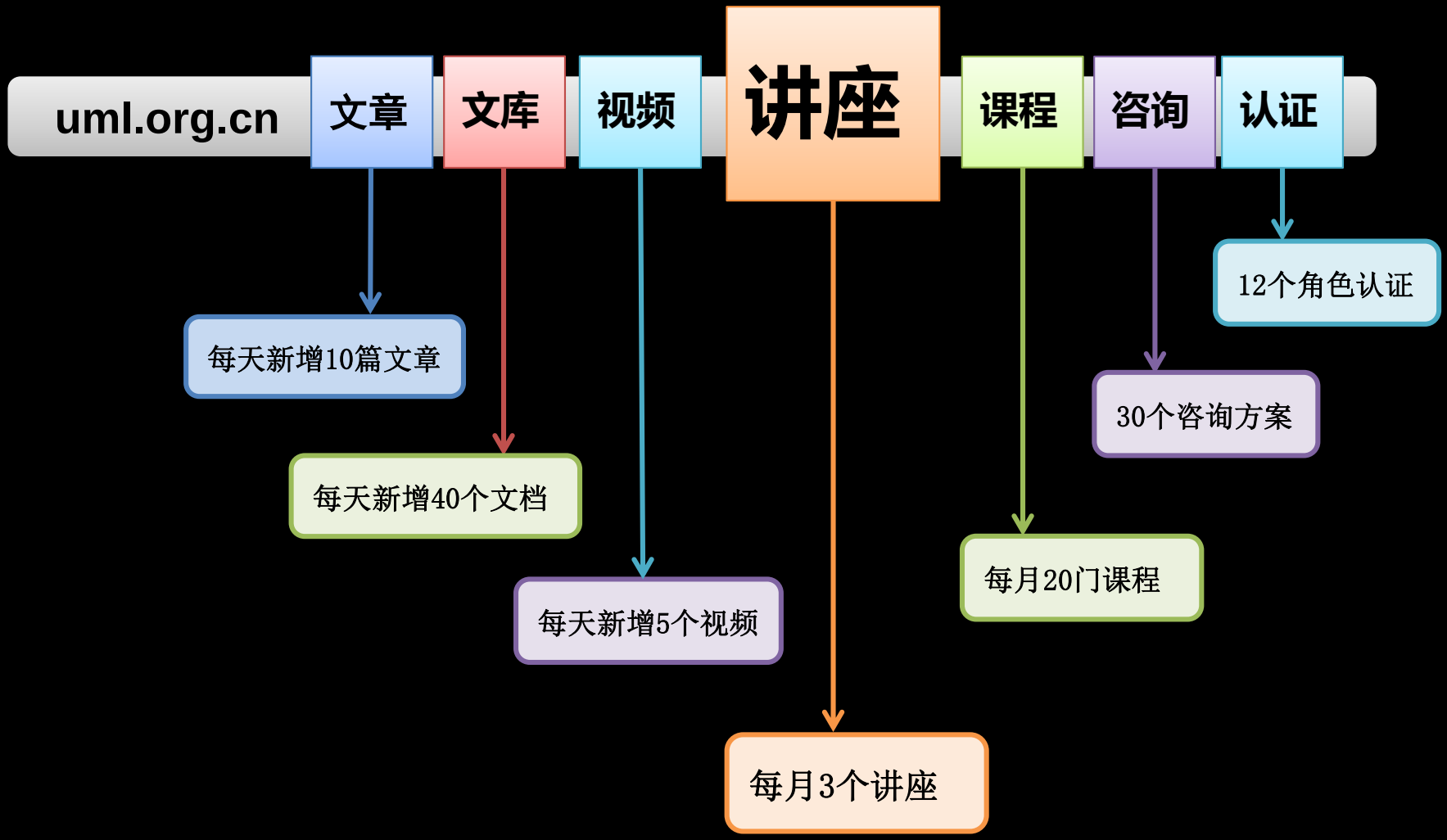
什么情况不用UML

- 简单、小规模、短期项目
- 不需要持续维护、建设的项目
- 不期望提高通用设计能力、只是专注某项技术的人
- 小规模团队，而且团队不会持续扩展

UML怎样才能学好

- 以产品为目标进行建模，力求：
  - 视角清晰、
  - 形式简单、
  - 内容明确
- 多临摹别人的作品
- 以可读性为目标
- 不断对照规范提升自己




uml.org.cn
文章
文库
视频
讲座
课程
咨询
认证
每天新增10篇文章
每天新增40个文档
每天新增5个视频
每月3个讲座
每月20门课程
30个咨询方案
12个角色认证

更多分享，更多成长！