

Table of Contents

前言	1.1
产品流程	1.2
需求采集	1.2.1
需求定义	1.2.2
迭代管理	1.2.3
PRD定义	1.2.4
需求变更	1.2.5
产品验收	1.2.6
研发流程	1.3
接口定义	1.3.1
HLD定义	1.3.2
开发定义	1.3.3
待发布定义	1.3.4
发布定义	1.3.5
线上维护	1.3.6
测试流程	1.4
测试用例	1.4.1
测试流程	1.4.2
环境管理	1.5
开发环境	1.5.1
代码环境	1.5.2
运维环境	1.5.3
测试环境	1.5.4
线上环境	1.5.5
管理环境	1.5.6
知识库环境	1.5.7
文件环境	1.5.8
附录	1.6
版本控制	1.6.1
迭代日历	1.6.2
分支管理	1.6.3
代码管理	1.6.4
产品定义	1.6.5
迭代定义	1.6.6
工具	1.7
E-R图	1.7.1
UML图	1.7.2

研发开发流程V1.0

研发在产品研发的过程中，需要有一定地工作节奏，此工作节奏具有一定地方法论特性，通过科学合理的调配开发顺序，制定开发章程，来打通软件研发过程中的工程化特性，减少协作中的不必要浪费，培养研发具备独立思考、提升个人能力等。

产品流程

本讲主要是针对产品部门进行相关的工作节奏梳理，产品部门的主要职责在于：

- 基于战略，采集、分析、定义市场需求
- 将定义好的市场需求 转化为 产品设计 输出给研发
- 在产品开发完成后，针对产品设计进行验收
- 针对定义的市场需求，通过必要的手段进行市场验证

产品团队主要是定义、验证产品方向，保证产品的价值，促成公司的商业价值。

需求采集

输入：

- 客户的反馈
- 产品经理的提案
- 关键干系人的提议

输出：

- 需求采集文档

注：

1. 1 x 产品 = n x 需求采集
2. 需求采集文档 以产品为单位
3. 需求采集文档必须进行定期维护，维护人员由产品经理定义
4. 需求采集文档编号配置规则：{year}{month}{date}{千位数字} 如：201901060001

需求采集文档：

大数据平台需求采集文档

编号	需求	描述	采集人	提报人	采集时间	采集状态	备注
201901060001	对接电信接口	我们的客户在获客过程中存在着两种获客方式，1、电信获客（0.3元一条）2、联通获客（1.2元一条）当天下已接通联通，但联通的反馈不是很好，很多客户还是选择回归到电信，所以，为了迎合客户，我们需要针对电信进行相关的对接操作	杨荣凯	孙克岗（boss）	2018年12月25日	已采纳	
201901060002	订单账务修正	运维反馈，基于订单账务问题，11月在平台产生的财务进账要算在12月份，因为12月份采用，而且，线下是赊账现象，一般用完结算	杨荣凯	肖杰（运营）	2019年1月5日	废弃	账务问题不能因为1时方便而致平台规范不顾
201901060002	坐席资源回收	运营反馈，基于有些客户已经购买了坐席，但后期不再使用，并通知到我们，我们可以有方式进行资源回收，再次卖给其他客户	杨荣凯	肖杰（运营）	2019年1月5日	审核中	

需求定义

输入：

- 需求采集文档

输出：

- 用户故事文档

用户故事 什么人，做什么事情，产生什么结果，有什么价值

注：

- 1 x 需求 = n (至少1个) x 用户故事
- 用户故事 必须满足条件
- 用户故事文档 必须进行定期维护，维护人员由产品经理定义
- 用户故事编号配置规则：{year}{month}{date}{千位数字} 如：201901060001
- 优先级定义范围为 (0~150)，值越小，级别越高，用于安排迭代

用户故事文档：

大数据用户故事

编号	角色	事件	结果	价值	优先级 (0~150)	需求采集编号	执行状态
201901060001	客户	购买电信坐席、数据	客户可以在商城中成功购买坐席产品	客户可以选择电信产品，给予选择权	30	201901060001	执行中
201901060002	客户	使用电信的数据、坐席	客户可以在客户端使用该产品	客户可以使用电信产品，降低成本	31	201901060001	执行中
201901060003	生产	可以进行电信的坐席、数据生产	生产方可以与电信对接成功，并可进行生产	添加新的供货渠道，提升企业生存	20	201901060001	完成
201901060004	生产	可以针对代理商设置电信坐席、数据的代理	生产方针对代理方提供产品代理	通过代理，扩大发展产品，促进客户生成	60	201901060001	完成
201901060005	生产	针对电信坐席、数据进行发货操作	生产方可以在发货时进行电信产品发货	服务客户，保证发货顺畅，高效	22	201901060001	完成
201901060006	代理	代理可以商城中设置电信产品	代理生可以在商城中上架电信产品	客户可以选择电信产品，给予选择权	61	201901060001	未执行
201901060007	代理	代理电信坐席、数据产品	代理电可以针对下级代理进行产品代理	代理可以选择代理电信产品，给予选择权	62	201901060001	未执行

迭代管理

输入：

- 用户故事

输出：

- 产品迭代计划书

注：

- 1 x 用户故事 = n (至少1个) x 任务项
- 产品迭代计划书 != 研发工期任务表
- 产品迭代计划书由产品经理管理及维护
- 产品迭代计划书必须包含 计划、执行、验收 三大跟进状态
- 在执行前，产品必须附上PRD及相关的产品原型、高保真，信息编写在备注里
- 在验收时，必须严格按照验收标准来执行。
- 迭代编号配置规则：{year}{month}{date}{千位数字} 如：201901060001

产品迭代计划书：

大数据产品迭代计划书

编号	用户故事编号	任务项	任务描述	工期预估 (h)	预计开发时间	预计完成时间	开发时间	完成时间	验收标准	验收时间	验收状态	计划人	计划时间	备注
201901060001	201901060003	电信坐席生产车间	完整针对电信的坐席对接工作	4	2019年1月7日	2019年1月15日	2019年1月7日		1. 可以正常的生产电信坐席 2. 可以在生产车间页面进行相关的展示，包含列表、统计及搜索 3. 添加坐席提取功能			史志鹏	2019年1月3日	
201901060002	201901060003	电信数据生产车间	完整针对电信的数据对接工作	4	2019年1月7日	2019年1月15日	2019年1月7日		1. 可以正常的生产电信数据 2. 可以在生产车间页面进行相关的展示，包含列表、统计及搜索 3. 添加数据提取功能			史志鹏	2019年1月3日	

PRD定义 (Product Requirement Document)

输入：

- 用户故事
- 迭代排期计划书

输出：

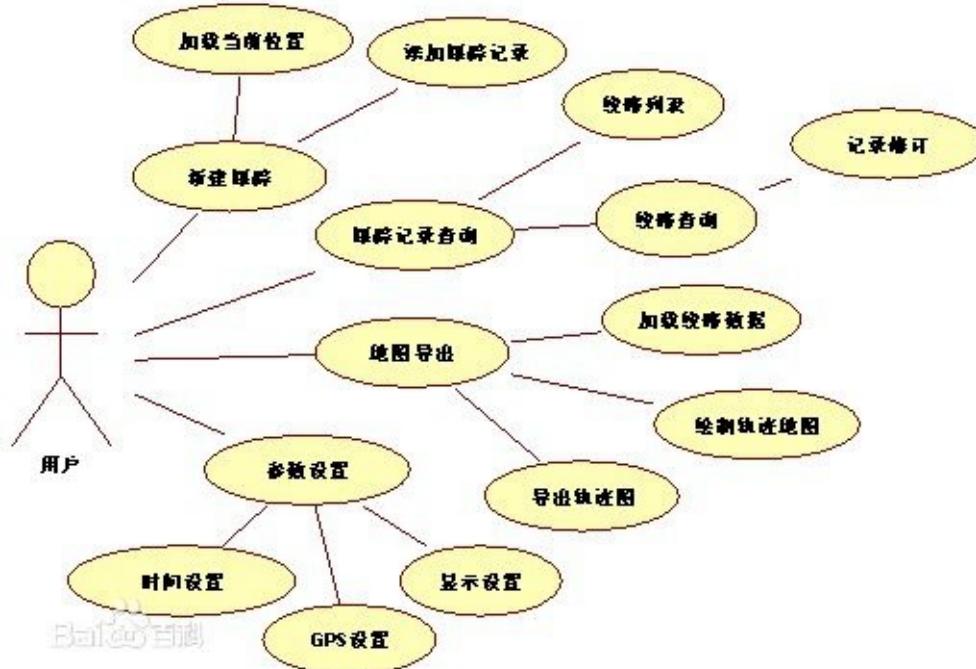
- PRD文档

PRD元素：

- 用例图
- 流程图
- 思维导图
- 线框图
- 原型稿

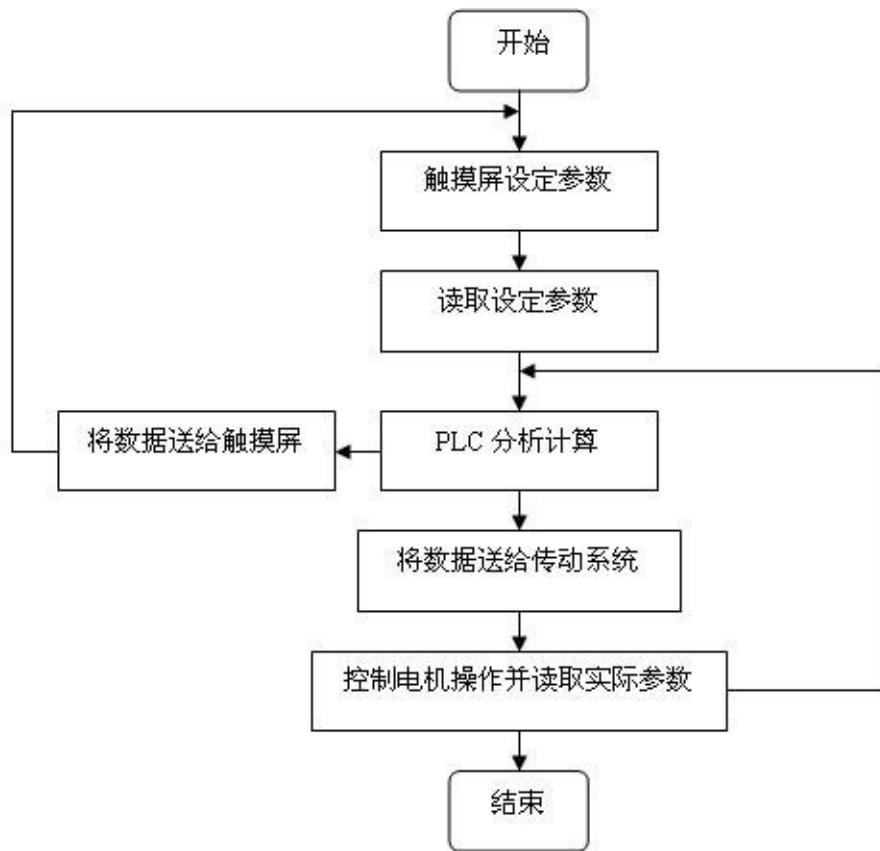
用例图

参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图



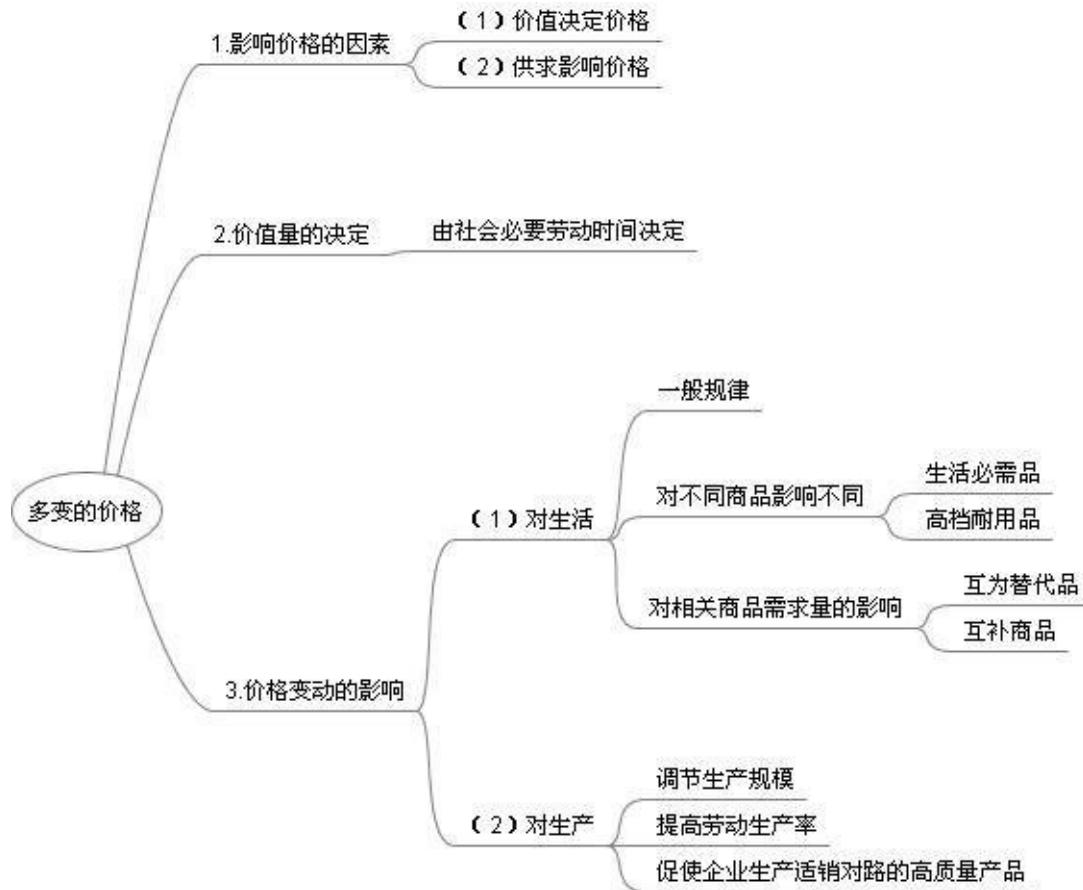
流程图

梳理需求流程，保证需求逻辑正确性。（活动、逻辑、方向）



思维导图

针对流程图中的活动，进行相关的分类操作，梳理具体的模块、功能、元素、动作等

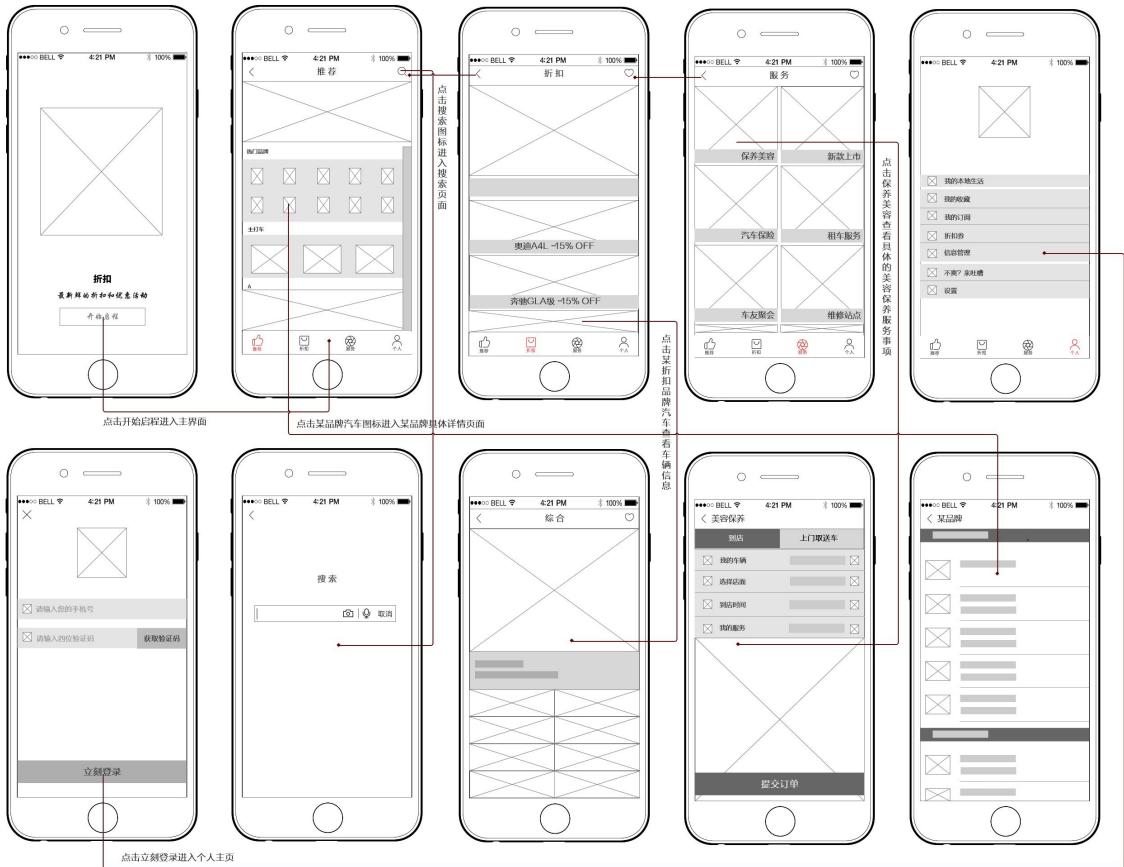


线框图

线框图是产品设计的低保真呈现方式。它有三个简单直接而明确的目标：

1. 呈现主体信息群
2. 勾勒出结构和布局
3. 用户交互界面的主视觉和描述

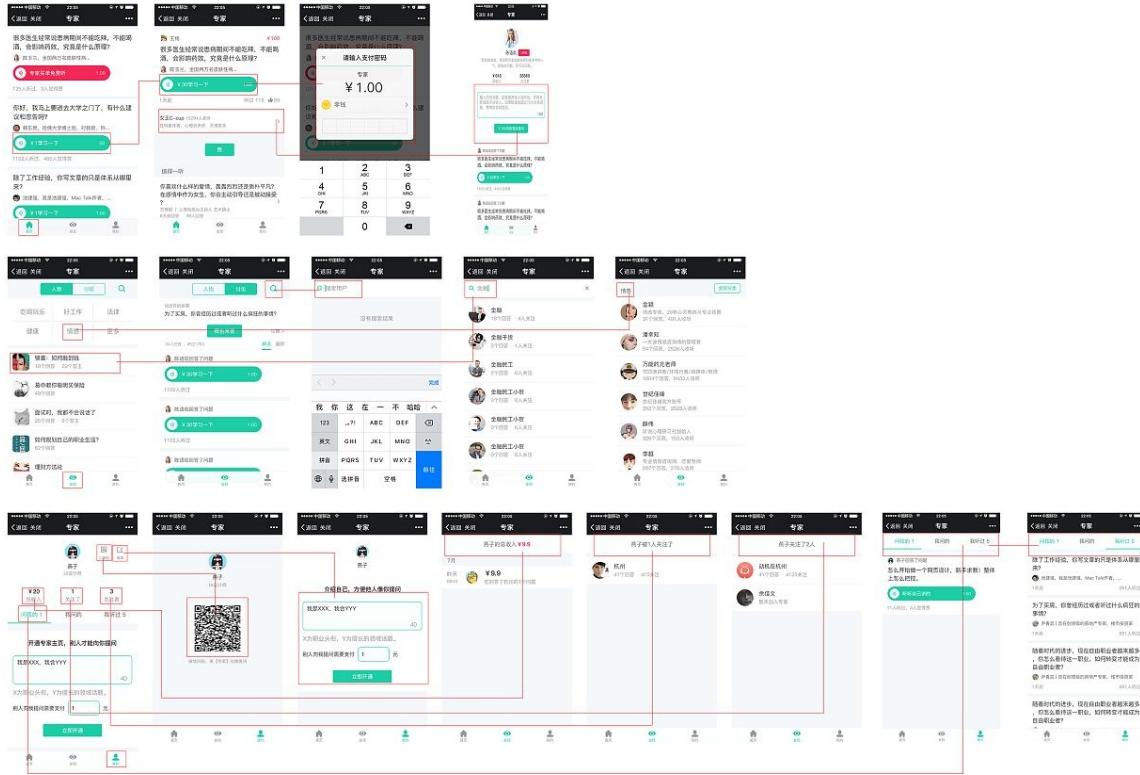
正确地创建了线框图之后，它将作为产品的骨干而存在。它就像一幢建筑的蓝图一样，将细节规定地明明白白。



原型稿

原型的要求比线框图/可交互式线框图要高，它要求必须是可交互的，并且尽可能贴合最终的用户界面的高保真模型。

制作原型的目标非常明确：尽可能真实地模拟用户和界面之间的交互。当一个按钮被按下的时候，相应的操作必须被执行，对应页面也必须出现，尽可能地模仿完整的产品体验。



需求变更流程

输入：

- PRD文档

输出：

- PRD文档
- 需求变更记录

产品验收 (Product Requirement Document)

输入：

- 用户故事
- 产品迭代计划书
- PRD文档

输出：

- 产品迭代计划书

产品验收

按照“产品迭代计划书”验收标准及PRD文档，进行相关的验收工作

研发流程

本讲主要是针对研发部门进行相关的工作节奏梳理，研发部门的主要职责在于：

- 基于产品的设计定义需求可行性
- 将具备可行的 产品设计 转化成相应的 产品功能
- 保证产品 数据、逻辑、功能 合理正确
- 保证产品 线上稳定性， 保证客户使用要求满意度

研发团队主要是针对已定义好的产品设计，评估、设计、实现、测试、部署、跟进。

接口定义

输入：

- PRD文档
- 产品原型（或高保真图）

输出：

- 接口文档
-

接口规范

- 协议规范
- 服务描述
- 错误标准
- 文件传输
- 签名算法
- 其他.....

接口定义

- 名称
 - 作者 (@author)
 - 版本 (@version)
 - 描述 (@desc)
 - 输入 (@param)
 - 输出 (@return)
-

Example

接口规范

请求协议头

公共参数	描述
flag	服务器标示
timestamp	时间戳
api	api标示
sign	协议签名
auth	当进行auth验证时，需要传该参数

返回报文

状态	返回
成功状态	<ul style="list-style-type: none"> status : 服务器状态 result : 返回api响应结果
失败状态	<ul style="list-style-type: none"> code : 服务器错误代码 status : 服务器状态 msg : 服务器错误消息

错误信息

错误类型	错误代码	错误描述
sys	10001	系统运行错误
pro	20001	数据被串改
pro	20002	协议参数丢失 {}
pro	20003	协议参数 {} 应为 {} 类型
pro	20004	数据访问超时
api	30001	请求的接口不存在
api	30002	请求频率超过上限
api	30003	请求接口超过时效性
api	30004	缺失参数{},请参考API文档
api	30005	参数格式不对{},请查阅API文档, 错误信息:{}
api	30006	参数{}值非法, 请参考API文档
api	30007	auth_token不存在
api	30008	auth_token已过期,请重新获取
api	30009	auth_token续签失败,请重新获取
business	40001	业务逻辑错误

系统提供服务列表

用户服务：针对注册登录用户提供服务

- [user – server](#) : django框架接收http协议

文件服务：提供文件上传服务

- [file – server](#) : django框架接收http协议

小程序服务：提供小程序服务

- [mini – server](#) : django框架接收http协议

系统提供API列表

用户服务（标识：user）：django框架接收http协议

api名称	api描述
<u>measure.staff.search</u>	员工绩效列表接口
<u>data.staff.resetstatus</u>	員工數據狀態重置接口
<u>user.staff.searchall</u>	员工列表接口
<u>shop.goods.match</u>	通过商品名称匹配商品基础信息
<u>data.mobileddevices.update</u>	手机设备导入数据修改接口
<u>product.productmodel.remove</u>	产品型号删除接口
<u>task.brokenpoint</u>	任务续传接口
<u>data.buyinfo.resetstatus</u>	購買訂單狀態重置接口

api签名算法

Created on 2016-7-11

@author: YRK

signature算法:

- 1、将所有非sign键的字典进行以键（string类型）反序排列，并以{key}{value}形式依次组装成字符串
- 2、将组装好的字符串进行sha1算法进行40长度字符串生成
- 3、通过参数数量决定获取抽样数量（默认抽样数量=参数数量*抽样因子（默认1.4）），并通过 签名长度 / 抽样次数 来决定间隔数量，通过抽样数量及间隔来决定抽取的字符串。

如：

```
原字典数据: { 'auth_token': '1231kandfkk1sdf', 'timestamp': 123154143123, 'method': 'get_user', 'account': 'test', 'passwd': 'test'}
生成字符串: timestamp123154143123passwdtestmethodget_userauth_token1231kandfkk1sdfaccounttest
算法生成: 114d8d89b3135639d75ed99144310274b7586299
最终采样: 1d19d47
```

注：如果 默认抽样数量 > sha1算法计算的签名长度，则立即返回sha1算法计算的签名

其他：

```
参数是 3个, 产生的密匙为: 11d7
参数是 4个, 产生的密匙为: 1bd4b
参数是 5个, 产生的密匙为: 1d37905
参数是 6个, 产生的密匙为: 1d19d478
参数是 7个, 产生的密匙为: 18b679348
参数是 8个, 产生的密匙为: 1d9135942b6
参数是 9个, 产生的密匙为: 1d816d13776
参数是 10个, 产生的密匙为: 14db337d940452
参数是 11个, 产生的密匙为: 14db16d59432b52
参数是 12个, 产生的密匙为: 14d91597d9417b82
参数是 13个, 产生的密匙为: 1488b3697d94304782
参数是 14个, 产生的密匙为: 1488b153d5914124782
参数是 15个, 产生的密匙为: 11dd933697ed94307b569
参数是 16个, 产生的密匙为: 11dd93153d5d914127b569
参数是 17个, 产生的密匙为: 11dd8b15697ed9431247569
参数是 18个, 产生的密匙为: 11d88b3353d7ed944107b7869
参数是 19个, 产生的密匙为: 11d88931569d5d9143127b7869
参数是 20个, 产生的密匙为: 1148d9b1353975d99441074b5829
参数是 21个, 产生的密匙为: 1148d8b33569d7ed9143107475829
```

文件上传

传输协议

url地址: http://[domain][:port]/file/upload

字段名	描述
auth_token	访问令牌
upload_file	文件IO流名称
store_type	上传文件类型, 如:

文件传输流HTTP协议剖析

```
POST /file/upload HTTP/1.1
Host: 192.168.1.240
Accept-Encoding: gzip, deflate
Connection: keep-alive
Accept: /*
User-Agent: python-requests/2.11.1
Content-Length: 626387
Content-Type: multipart/form-data; boundary=dbc716d9b20c4c5989d24951dae9d58b

--dbc716d9b20c4c5989d24951dae9d58b
Content-Disposition: form-data; name="store_type"

nick
--dbc716d9b20c4c5989d24951dae9d58b
Content-Disposition: form-data; name="auth_token"

7f0a14bdae056193
--dbc716d9b20c4c5989d24951dae9d58b
Content-Disposition: form-data; name="upload file"; filename="test.jpg"
```

方法名： equipment.equipmentout.search

```
@author : fsy
@version : v1.0

@desc : 设备出库数据列表接口

@param : search_info - dict # 搜索条件
{
    # 搜索条件
    agent_phone : char # 代理商电话 - (选参)
    product_type : char # 产品类型 - (选参)
    max_number : char # 终止号段 - (选参)
    add_time : date # 添加时间 - (选参)
    agent_name : char # 代理商名称 - (选参)
    type : char # 出库类型 - (选参)
    product_model : char # 产品型号 - (选参)
    min_number : char # 起始号段 - (选参)
}
@param : current_page - int # 当前查询页码

@return : data_list - list # 客户返利数据列表
[
    # 客户返利数据列表
    max_number : char # 终止号段
    remark : char # 备注
    create_time : datetime # 创建时间
    product_type : char # 产品类型
    rate : char # 签约费率
    id : int # id
    min_number : char # 起始号段
    agent_phone : char # 代理商电话
    agent_name : char # 代理商名称
    quantity : int # 入库数量
    price : char # 单价
    add_time : date # 添加时间
    type : char # 出库类型(self:自营平台,first:一级代理,second:二级代理)
    product_model : char # 产品型号
    salesman : char # 业务员
    address : char # 发货地址
}
.....
]

@return : total_page - int # 总页码数
@return : total - int # 数据总数
```

HLD定义 (High Level Design)

输入：

- PRD文档
- 产品原型（或高保真图稿）
- 接口文档

输出：

- HLD文档
-

HLD文档

要求：

- 数据库设计
- 逻辑设计
- 技术选型

数据库设计

团队需根据需求优先定义数据结构组成，此处必须经过项目负责人及架构师确认。可使用E-R图或数据库视图。

逻辑设计

团队需要学习UML建模语言，依托于逻辑设计来进行相关的开发前设计操作（强制）。类图，时序图严格执行。

技术选型

技术选型定义由架构师定义，技术推广需要架构师与项目负责人一起推行，架构师必要时需要进行相关的技术培训

工具

略

UML建模语言参考 - 工具->UML图

开发定义

输入：

- PRD文档
- 产品原型（或详情稿）
- 接口文档
- HLD文档
- 测试用例（自身）

输出：

- 持续集成版本
-

持续集成

集成规则：

- 每日需进行一次持续集成
- 持续集成时，每次提交必须是可运行版本，不得影响其他团队成员开发
- 持续集成暂定每天17:30进行
- 持续集成时，需要针对代码进行自动化检测，再进行部署
- 持续集成时，如存在自动化测试，则进行自动化测试

版本控制：

- 持续集成的版本暂定为dev版本
- 开发过程中，使用git版本管理
- git版本管理中，要学会使用future分支
- future开发完成后，再进行集成，不允许未开发完进行集成现象
- future分支需要进行了基础的单元测试才能提交

补充信息

集成好处 1、减少风险

- 尽早发现缺陷并修复缺陷；软件开发中每天持续集成，并进行测试和评审，这些过程有许多的机会发现缺陷。
- 尽早估量软件质量；通过在持续集成中实施持续测试和评审的活动，软件产品的健康属性，例如复杂性，将被全程跟踪。
- 尽早排除假设；通过在一个纯净的环境中，用同样的脚本和过程构建和测试软件，开发人员可以减少假设：是否使用了不正确的第三方的库，是否使用了不同的环境参数等。

2、减少重复的过程

- 任何时候都可以执行一致的流程；
- 每个开发过程的步骤都是依次执行的；在构建脚本，将依次执行编译，自动测试等动作。
- 在版本控制库中，当代码提交发生时，构建过程自动执行。通过减少在重复性工作上的劳动力，让开发工程师做更多有价值的工作。

3、产生可部署的软件

- 持续集成可以让项目组在任一点上及时提交可以安装的软件包。这是持续集成最可看见的一个益处。我们可以无休止地讨论改善软件质量和减少风险，但是对用户或者客户来说，可以安装的软件包是最切实可行的

4、使得项目更加透明

5、建立项目信心

集成阻碍 1、用于维护持续集成的费用比较高 这通常是一个被误导的错误的理解，如论你是否使用持续集成，你依然需要集成，构建，测试，部署等工作。管理一个强壮的持续集成的系统比管理手动的过程更加高效率。

2、太多的变更 一些开发组织需要改变已有的开发习惯和过程。

3、太多失败的构建 当开发人员不做个人构建之后，就将变更的代码提交到版本控制库，从而导致太多的构建失败。

4、额外的软件和硬件成本 持续构建需要独立的集成构建主机和持续集成软件等，这些需要额外的成本。

待发布定义

输入：

- PRD文档
- 产品原型（或详情稿）
- 接口文档
- HLD文档
- 测试用例
- 软件集成测试版本

输出：

- 软件待发布版本
 - 软件测试报告
 - 待发布文档（包含操作步骤、回滚操作、文案说明、升级公告、内部公告等）
 - 脚本准备（数据迁移、升级，结构修改等）
-

软件待发布版本

- 采用版本控制 release分支，参考附录-版本控制
- 版本需要给到测试进行最后的上线前发布测试
- 待发布版本需要上线模拟，至少3次，保证发布顺畅

软件测试报告

- alpha软件测试报告
- beta软件测试报告
- release软件测试报告

待发布文档

- 操作步骤（jenkins）
- 回滚操作（jenkins）
- 内部公告（产品出、研发执行）
- 升级公告（产品出、研发执行）

脚本准备：

- 结构升级脚本
- 数据升级脚本
- 自动化部署、发布脚本

发布定义

输入：

- 软件待发布版本
- 待发布文档（包含回滚定义）
- jenkins准备工作完成

输出：

- 线上可运行版本
 - 发布记录文档（什么时间点开始，什么时间点结束，发布人员，发布结果，包好文案及注意事项）
 - 公司内部及客户公告
-

线上可运行版本

- 经历过beta、release测试的版本
- 经历过产品验收的版本
- 该版本必须添加 tag，详见附录-版本控制

发布记录文档

- 什么时间点开始
- 什么时间点结束
- 发布人员
- 发布结果
- 发布文案
- 注意事项

注：此部分项目经理负责，告知产品

公司内部及客户公告

- 发布完成后，要针对关键干系人进行通知公告

线上维护

输入：

- 线上版本

输出：

- bug补丁版本
-

bug补丁版本

- bug补丁必须在master分支上，并学会用bugfix分支使用
- bug补丁修复完成后要同步到dev分支上，同时，要保证dev可以正常运行
- bug补丁在发布时，需要进行发布流程，并添加发布记录（项目经理负责，通知产品）
- bug补丁版本完成后，要按照附录-版本控制添加tag

bug补丁原则：紧急的bug立马修，非紧急的bug迭代修

研发流程

本讲主要是针对测试部门进行相关的工作节奏梳理，研发部门的主要职责在于：

- 基于产品的设计定义需求及研发实现的功能进行相关的质量检测
- 针对产品的整体流程、逻辑、功能进行全面的跟踪
- 确保上线前的稳定性，交付产品部验收的版本

测试团队主要是针对已定义好的产品设计及软件实现，针对性的做上线前的产品保证，是客户使用前的第一道屏障，该团队从黑盒、白盒、稳定性、压力等方向上都需要对其进行一定的监控，尽可能的保证万无一失。

测试用例

输入：

- PRD文档
- 产品原型（或详情稿）
- 接口文档

输出：

- 测试用例

测试流程

输入：

- PRD文档
- 产品原型（或详情稿）
- 测试用例
- 接口文档
- 持续集成版本

输出：

- alpha测试报告
- bata测试报告
- release测试报告

开发环境

环境简述：

狭义的公共开发环境主要是针对研发成员在开发过程中使用的公共环境，比较好理解的如下：

- 数据库环境（mysql、sqlserver、oracle、mongodb）
- 缓存环境（memcache、redis）
- 消息队列环境（rabbitmq、activemq）
- 日志环境（elk）

广义的公共开发环境不止包含上述环境，还可包含：

- 常用架构中间件（如上）
 - 自身编写的服务（系统自身封装的服务或中间的）
-

环境清单：

服务项	服务器地址	端口	账号	密码	服务描述
redis	192.168.3.250	6379			提供redis的缓存服务，redis具备多数据结构支持，同时提供数据持久化服务
mysql	192.168.3.250	3306	root	123456	提供redis的缓存服务，redis具备多数据结构支持，同时提供数据持久化服务

代码环境

环境简述：

- 针对研发在生产过程中的代码管理。

代码类型：

- 产品实际部署代码
- 研发技术沉淀代码

使用场景：

- 项目初始化拼接使用
- 研发开发过程使用
- 测试发布过程使用
- 运维上线过程使用

工具名称	服务器地址	端口	账号	密码	服务描述
gitlab	192.168.3.249	10080			代码管理服务器

运维环境

环境简述：

- 可以控制线上、测试环境的安装、部署、调试环境。

配备规则：

- 多个产品尽可能使用1套运维工具

使用场景：

- 进行持续集成
- 进行测试环境安装
- 进行测试版本发布
- 进行线上环境安装
- 进行线上版本发布
- 定时的备份，维护，监控机器

工具名称	服务器地址	端口	账号	密码	服务描述
jinkens	192.168.3.65	8084	admin	P@ssw0rd	提供redis的缓存服务，redis具备多数据结构支持，同时提供数据持久化服务

测试环境

环境简述：

- 针对研发开发结果进行相关的验收环境。

配备规则：

- 1个项目组配备1个测试环境
- 1个测试环境可基于项目组承载多个开发项目

使用场景：

- 进行持续集成
 - 进行发布前版本测试
 - 进行待发布模拟操作
 - 进行线上bug检测
-

注：

1. 基于成本考虑，必圈的测试环境以在线环境为主。
2. 测试环境部署采用虚拟化技术，当前使用 docker 容器技术。

线上环境

环境简述：

- 提供给客户使用的环境。

配备规则：

- 1个项目组至少有1个产品
- 1个产品配备至少1个线上环境

使用场景：

- 线上为客户提供服务
 - 线上为运维提供支撑
-

注：

1. 基于成本考虑，必圈的线上环境以在线环境为主。
2. 线上环境部署采用虚拟化技术，当前使用 docker 容器技术。

管理环境

环境简述：

- 针对需求、用例、bug等管理的环境

配备规则：

- 多个产品使用1套管理工具

使用场景：

- 需求提报、验收
- 用例编写、验收
- bug提报、验收

工具名称	服务器地址	端口	账号	密码	服务描述
禅道	192.168.3.65	8082	admin	P@ssw0rd	工作流程工具

知识库环境

环境简述：

- 研发过程中的规范、章程、培训课件、知识积累等存储点

配备规则：

- 研发团队仅有1套知识库环境管理

使用场景：

- 公司、研发章程、制度
- 研发团队工作流中的规范、约束 (ui、开发、测试、产品、架构、质检等)
- 研发培训资料放置点
- 研发知识分享点

服务名	服务器地址	端口	账号	密码	服务描述
Confluence	192.168.3.65	8087	admin	P@ssw0rd	共享知识库

针对文件存储环境（软件、文档、UI稿啊等等）

文件环境

环境简述：

- 针对研发过程性的文案进行归档

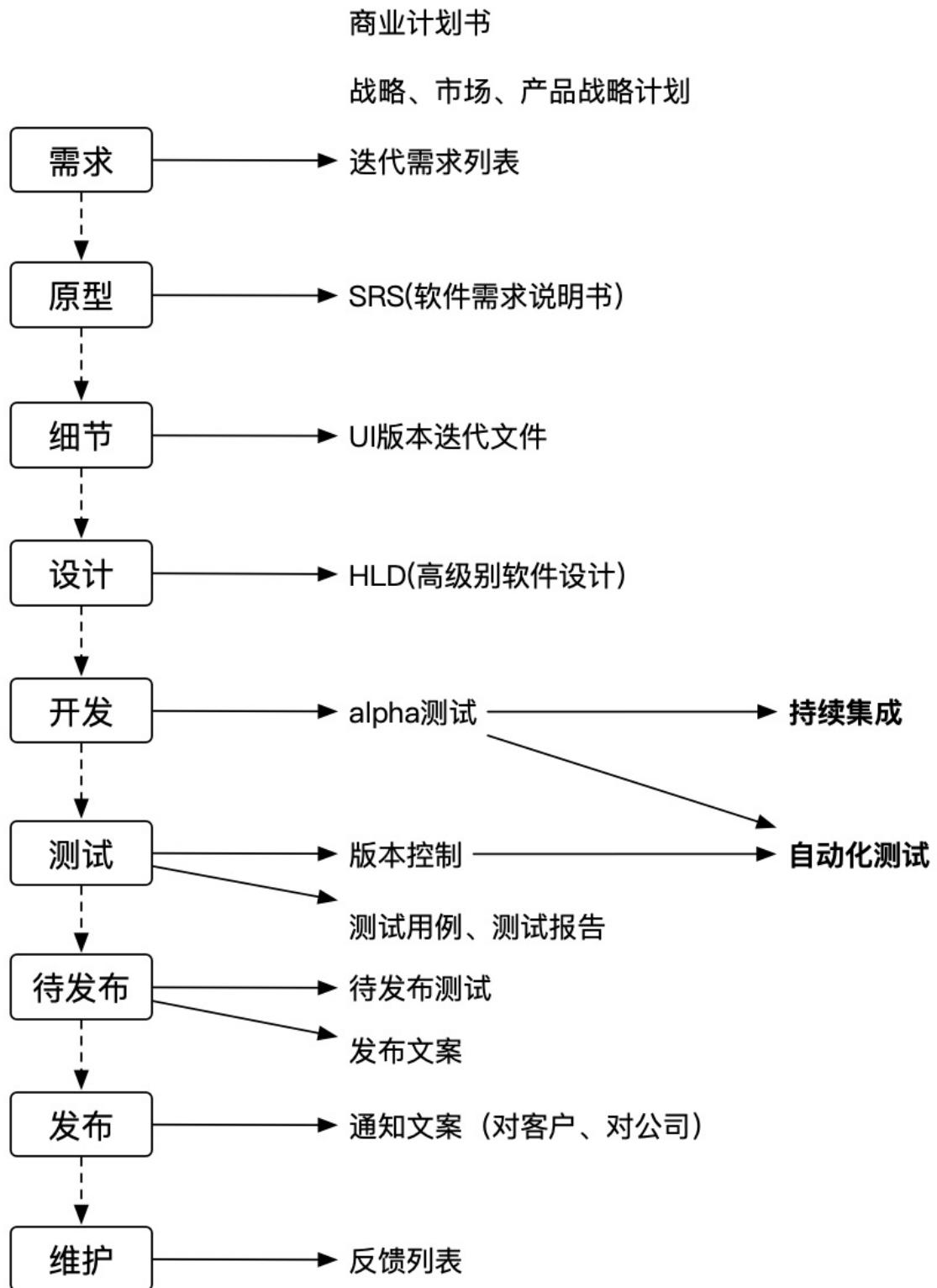
配备规则：

- 研发团队仅有1套文件存储环境

使用场景：

- UI、开发、测试、产品、架构、质检工作过程中归档性文档
- 研发公共使用的临时性文件
- 研发公共使用的软件库

服务器	服务器地址	端口	账号	密码	服务描述
samba	192.168.3.65				文件管理服务



版本管理

[主版本号] . [次版本号] . [补丁版本号] . [发布版本] . [集成版本号]

主版本号（产品定义）

主要用于大的版本迭代或大型的逻辑、数据、代码重构之用，发布时 +1；范围 00~99。

次版本号（研发定义）

主要用于需求、功能的迭代之用，每经过一个小范围需求或功能发布时 +1；范围 00~99。

补丁版本号（研发定义）

主要用于线上存有不得不解决的bug时，临时进行的线上修复，当补丁发布时 +1；范围 00~99。

发布版本（研发定义）

主要存在3中版本，分别为：alpha（单元测试版本）、beta（集成测试版本）、release（待发布版本）。

集成版本号（测试定义）

该版本主要是配合“发布版本”，在每次打出“发布版本”时，则自动 +1；范围 00~99。

研发日历



研发日历

	周一	周二	周三	周四	周五
上午		计划会议	例会	例会	例会
下午		回顾会议（可选）			
晚上	发布、通知				

UI日历 (2.5天下个迭代原型设计 + 1.5天本次迭代UI测试 + 半天任务计划 + 半天思考)

	周一	周二	周三	周四	周五
上午	ui测试	计划会议	例会	例会	例会
下午	ui测试	回顾会议（可选）	原型评审?	原型评审 细节评审?	ui测试
晚上	发布、通知				

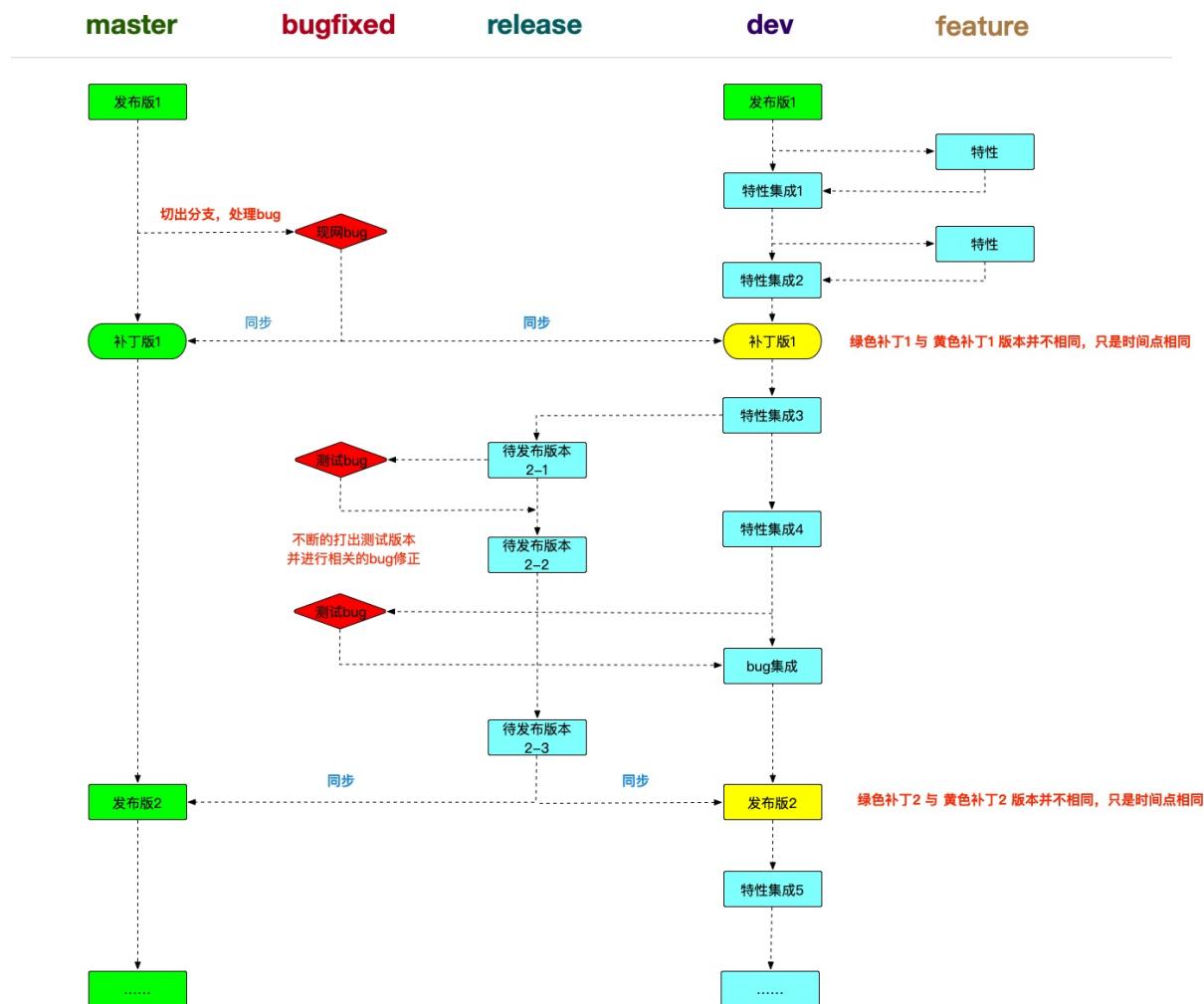
开发日历 (3天设计开发 + 1天测试、发布准备 + 半天任务计划 + 半天思考)

	周一	周二	周三	周四	周五
上午	模拟线上发布 bate测试 完成release版本打出	计划会议	例会	例会	例会
下午	release测试 上线准备	回顾会议（可选）	单元测试 持续集成 打出单元测试版本	单元测试 持续集成 打出单元测试版本	单元测试 持续集成 打出集成测试版本
晚上	发布、通知				

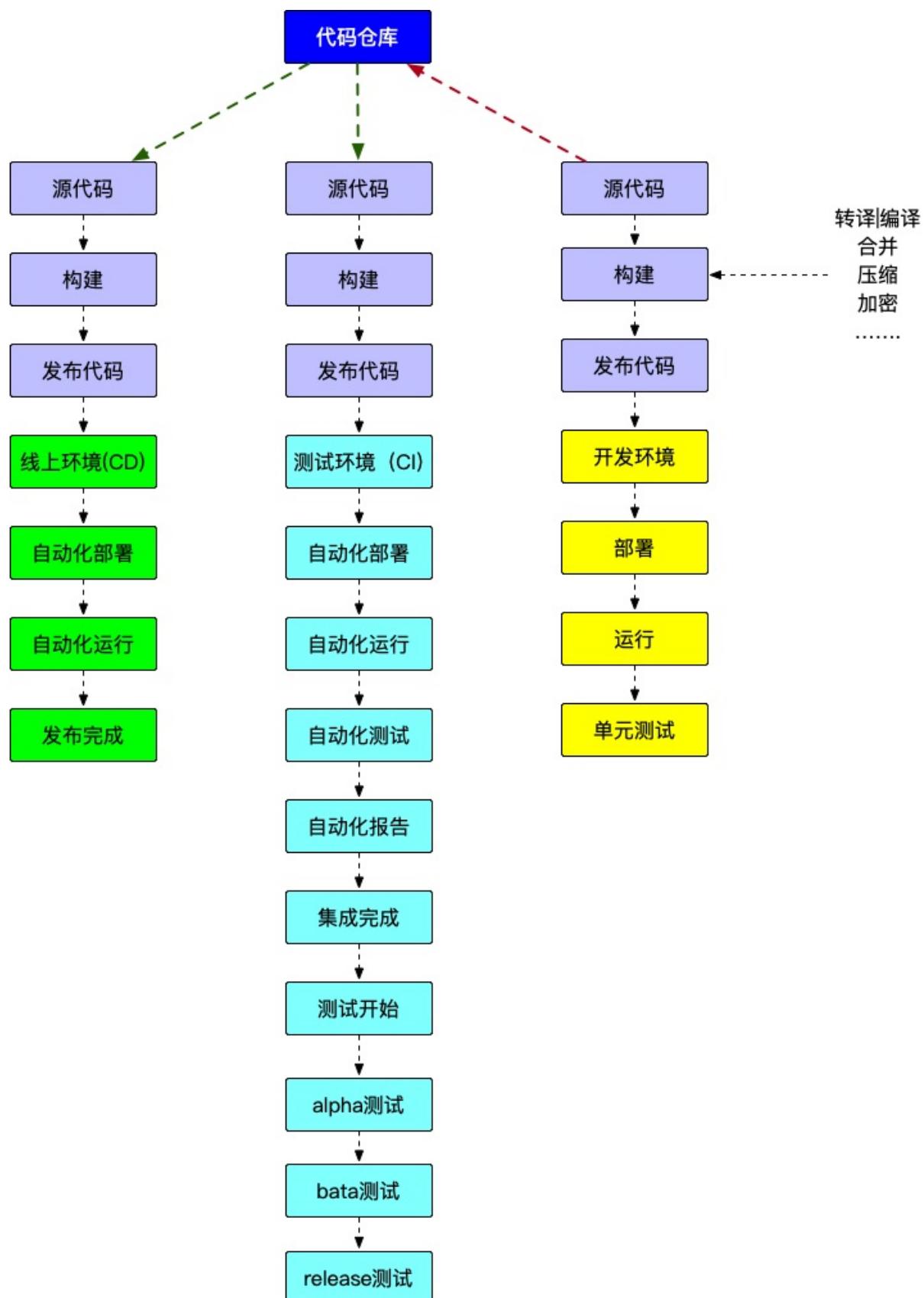
测试日历 (1天用例 + 3天测试 + 半天任务计划 + 半天思考)

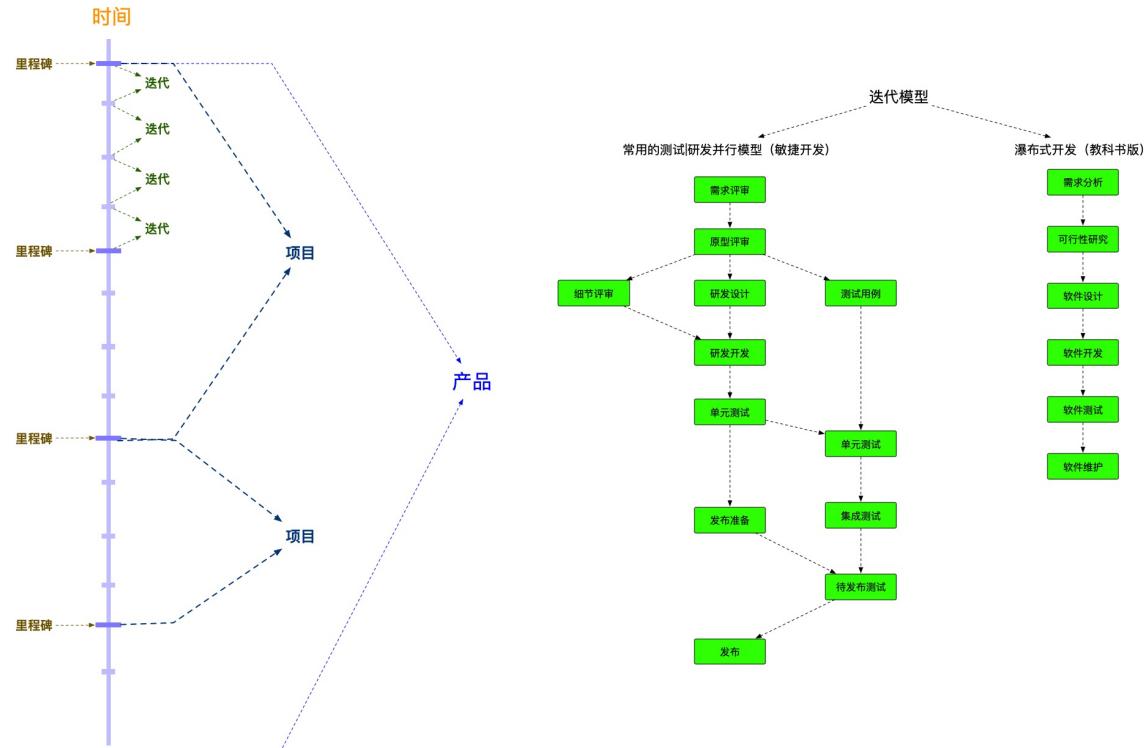
	周一	周二	周三	周四	周五
上午	bate测试、跟踪、回归	计划会议	例会	例会	例会
下午	release测试、跟踪、回归	回顾会议（可选）	用例编写 版本打出监督	单元测试报告、跟踪、回 归	单元测试报告、跟踪、回 归
晚上	发布、通知				

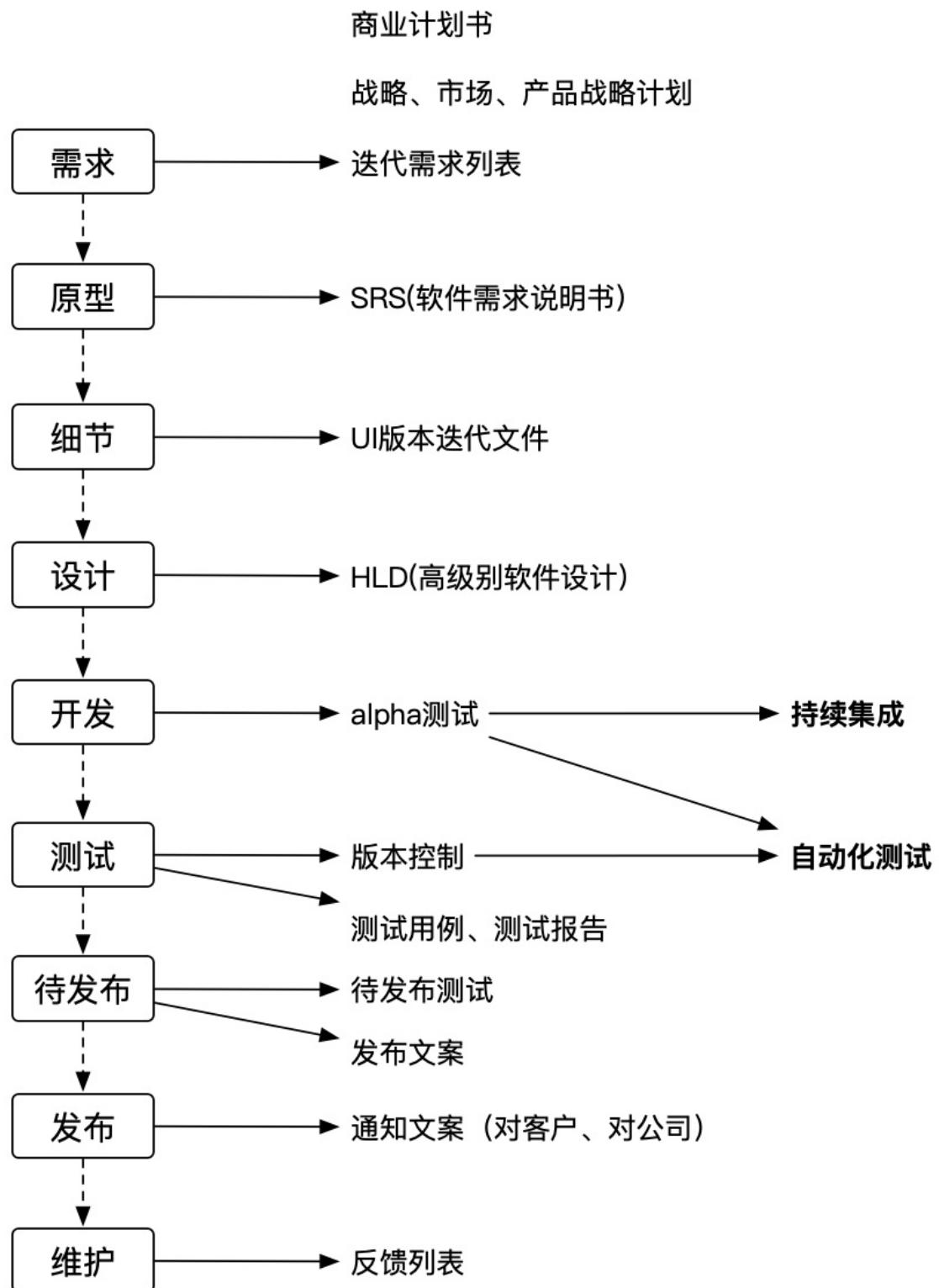
分支管理



代码管理







E-R图

ER图分为 实体、属性、关系 三个核心部分。

实体是长方形， 属性是椭圆形， 关系为菱形。

实体 (entity) :

即数据模型中的数据对象（即数据表），用长方体来表示，每个实体都有自己的实体成员（entity member）或者说实体对象（entity instance），例如学生实体里包括张三、李四等。

属性 (attribute) :

即实体所具有的属性，例如学生具有姓名、学号、年级等属性，用椭圆形表示，属性分为唯一属性（unique attribute）和非唯一属性，唯一属性指的是唯一可用来标识该实体实例或者成员的属性，用下划线表示，一般来讲实体都至少有一个唯一属性。

关系 (relationship) :

用来表现数据对象与数据对象之间的联系，例如学生的实体和成绩表的实体之间有一定的联系，每个学生都有自己的成绩表，这就是一种关系，关系用菱形来表示。

关联关系有三种：

1对1 (1:1) :

指对于实体集A与实体集B，A中的每一个实体至多与B中一个实体有关系；反之，在实体集B中的每个实体至多与实体集A中一个实体有关系。

1对多 (1:N) :

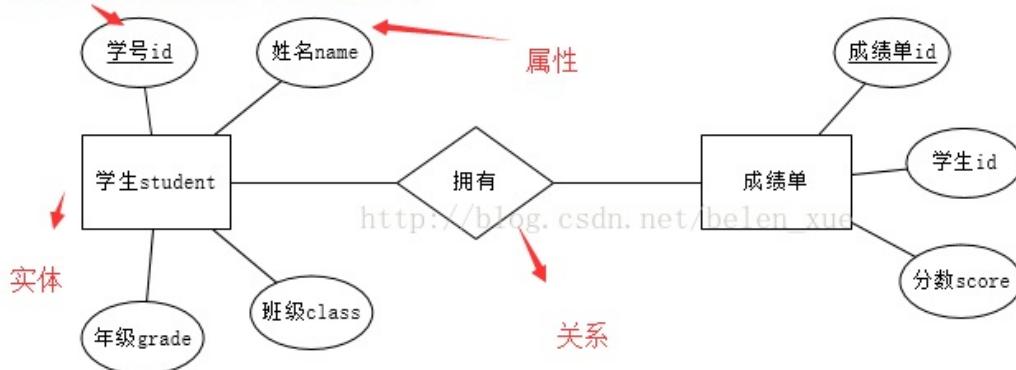
1对多关系是指实体集A与实体集B中至少有N(N>0)个实体有关系；并且实体集B中每一个实体至多与实体集A中一个实体有关系。

多对多 (M:N) :

多对多关系是指实体集A中的每一个实体与实体集B中至少有M(M>0)个实体有关系，并且实体集B中的每一个实体与实体集A中的至少N (N>0) 个实体有关系。

举例说明：

作为主键的属性在文字下面加下划线



<https://blog.csdn.net/zgcr654321>

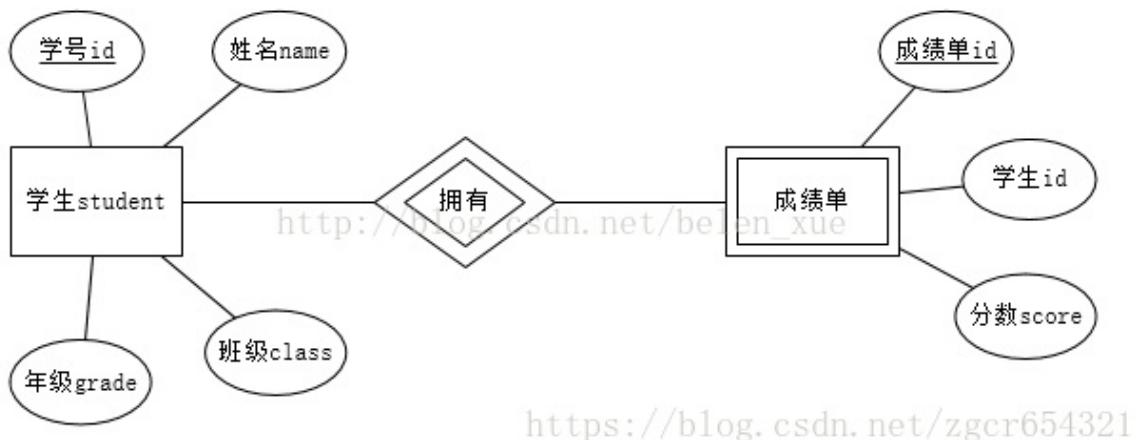
ER实体补充讲解：

ER的实体还会细分为弱实体和复合实体：

弱实体：一个实体必须依赖于另一个实体存在，那么前者是弱实体，后者是强实体，弱实体必须依赖强实体存在，例如上图的学生实体和成绩单实体，成绩单依赖于学生实体而存在，因此学生是强实体，而成绩单是弱实体。

弱实体和强实体的联系必然只有1: N或者1: 1，这是由于弱实体完全依赖于强实体，强实体不存在，那么弱实体就不存在，所以弱实体是完全参与联系的，因此弱实体与联系之间的联系也是用的双线菱形。

上面实例根据弱实体的情况更改如下图：



<https://blog.csdn.net/zgcr654321>

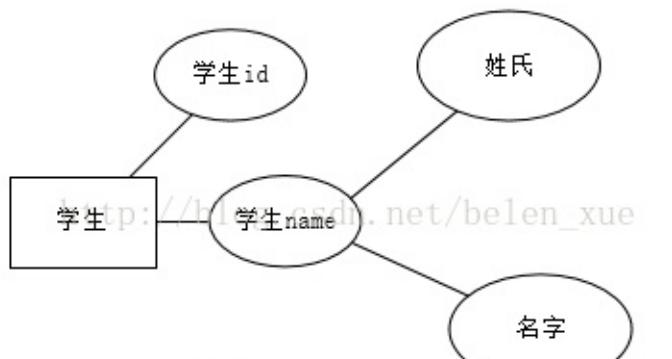
复合实体：复合实体也称联合实体或桥接实体，常常用于实现两个或多个实体间的M: N联系，它由每个关联实体的主要组成，用长方体内加一个菱形来表示。

ER属性补充讲解：

ER图的属性还细分为复合属性、多值属性和派生属性、可选属性，同时还有用来表示联系的属性，称为联系属性；

复合属性(composite attribute)：

复合属性是指具有多个属性的组合，例如名字属性，它可以包含姓氏属性和名字属性，如下图：

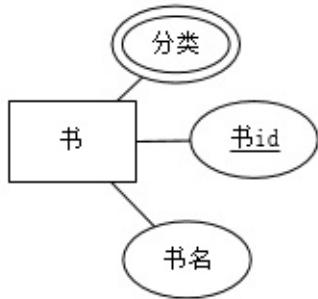


<https://blog.csdn.net/zgcr654321>

复合属性也有唯一属性，例如学生的所在班级属性，由于多个年级都有班级，所以单单班级属性是不唯一的，但是和年级组成的复合属性后则可以匹配成唯一属性。

多值属性 (multivalued attribute)：

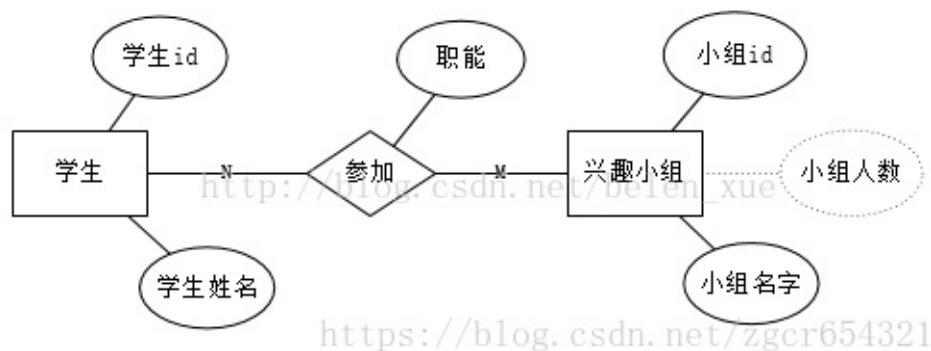
一个实体的某个属性可以有多个不同的取值，例如一本书的分类属性，这本书有多个分类，例如科学、医学等，这个分类就是多值属性，用双线椭圆表示。



派生属性(deriviers attribute):

是非永久性存于数据库的属性。派生属性的值可以从别的属性值或其他数据（如当前日期）派生出来，用虚线椭圆表示，如下图。

下面的小组人数就是典型的派生属性，随着学生实例的参加的兴趣小组变化，小组人数属性也会变化，一般来讲派生属性不存在于数据库中，而是通过相应的公式进行计算得到，如果要放到数据库中，那么隔一段时间就要进行更新，否则会出现数据错误。



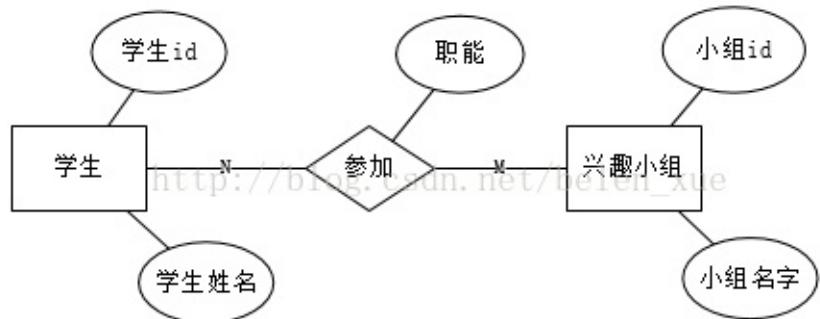
可选属性(optional attribute):

并不是所有的属性都必须有值，有些属性的可以没有值，这就是可选属性，在椭圆的文字后用(O)来表示，如下图的地址就是一个可选属性。



联系属性:

联系属于用户表示多个实体之间联系所具有的属性，一般来讲M:N的两个实体的联系具有联系属性，在1:1和1: M的实体联系中联系属性并不必要。



<https://blog.csdn.net/zgcr654321>

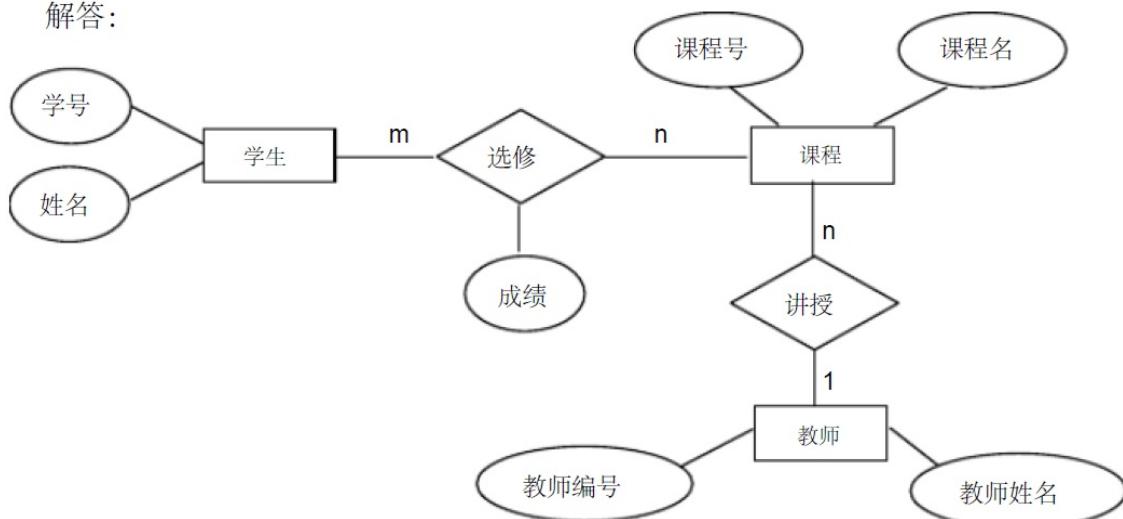
实例题目：

假设教学管理规定：

一个学生可选修多门课，一门课有若干学生选修；一个教师可讲授多门课，一门课只有一个教师讲授；一个学生选修一门课，仅有一个成绩。学生的属性有学号、学生姓名；教师的属性有教师编号，教师姓名；课程的属性有课程号、课程名。

要求：根据上述语义画出ER 图，要求在图中画出实体的属性并注明联系的类型；

解答：



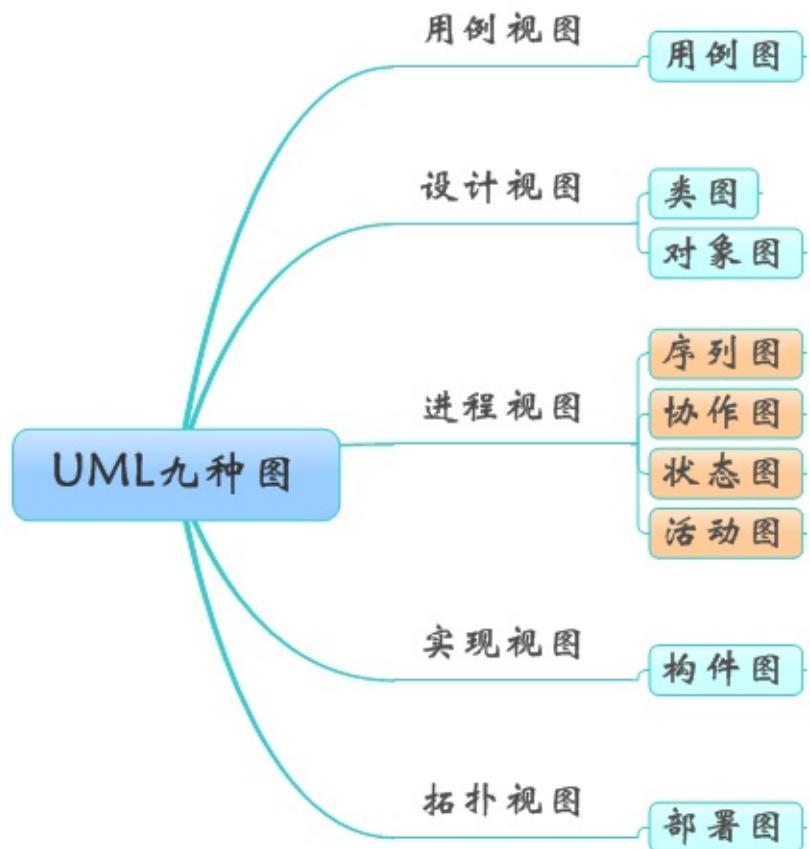
<https://blog.csdn.net/zgcr654321>

UML图

UML (Unified Modeling Language) 是一种统一建模语言，为面向对象开发系统的产品进行说明、可视化、和编制文档的一种标准语言。下面将对UML的九种图+包图的基本概念进行介绍以及各个图的使用场景。

一、基本概念

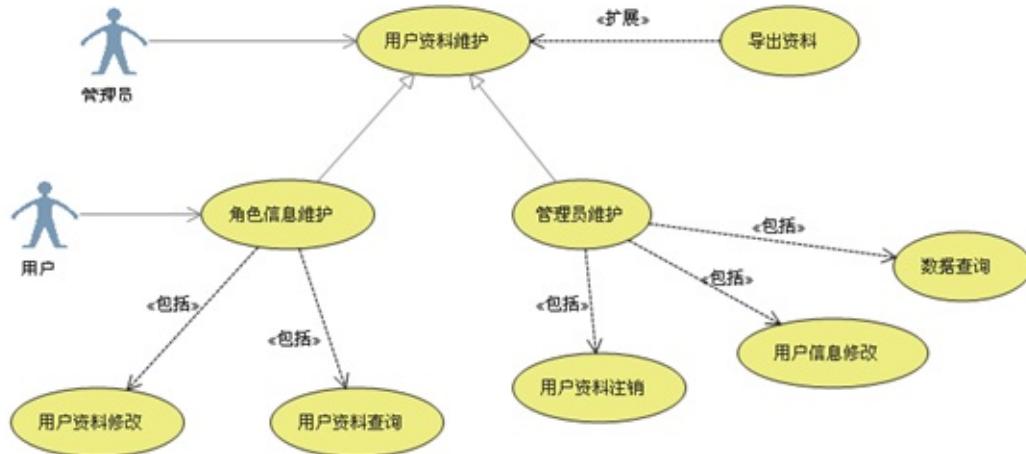
如下图所示，UML图分为用例视图、设计视图、进程视图、实现视图和拓扑视图，又可以静态分为静态视图和动态视图。静态图分为：用例图，类图，对象图，包图，构件图，部署图。动态图分为：状态图，活动图，协作图，序列图。



http://blog.csdn.net/zhou2s_101216

1、用例图 (UseCase Diagrams) :

用例图主要回答了两个问题：1、是谁用软件。2、软件的功能。从用户的角度描述了系统的功能，并指出各个功能的执行者，强调用户的使用者，系统为执行者完成哪些功能。



2、类图 (Class Diagrams) :

用户根据用例图抽象成类，描述类的内部结构和类与类之间的关系，是一种静态结构图。在UML类图中，常见的有以下几种关系：泛化 (Generalization)，实现 (Realization)，关联 (Association)，聚合 (Aggregation)，组合 (Composition)，依赖 (Dependency)。

各种关系的强弱顺序：泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖

2.1. 泛化 【泛化关系】：是一种继承关系，表示一般与特殊的关系，它指定了子类如何继承父类的所有特征和行为。例

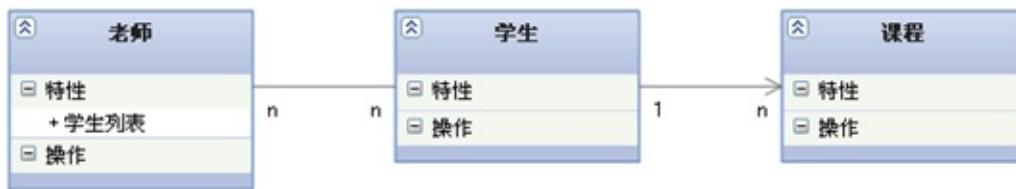


如：老虎是动物的一种，即有老虎的特性也有动物的共性。

2.2.实现 【实现关系】：是一种类与接口的关系，表示类是接口所有特征和行为的实现。

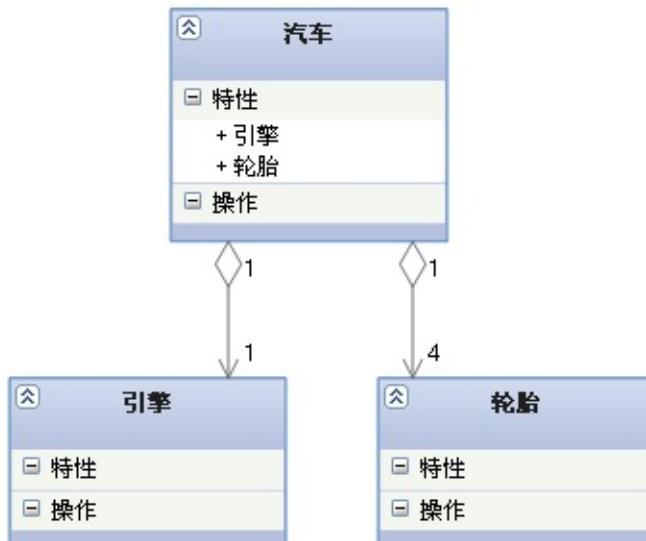


2.3.关联 【关联关系】：是一种拥有的关系，它使一个类知道另一个类的属性和方法；如：老师与学生，丈夫与妻子关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。



2.4.聚合 【聚合关系】：是整体与部分的关系，且部分可以离开整体而单独存在。如车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。

聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。

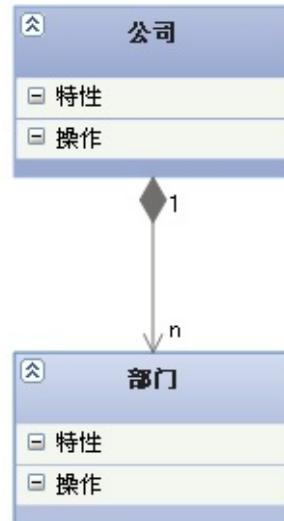


2.5.组合

【组合关系】：是整体与部分的关系，但部分不能离开整体而单独存在。如公司和部门是整体和部分的关系，没有公司就不存在部门。

组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

【代码体现】：成员变量



【箭头及指向】：带实心菱形的实线，菱形指向整体

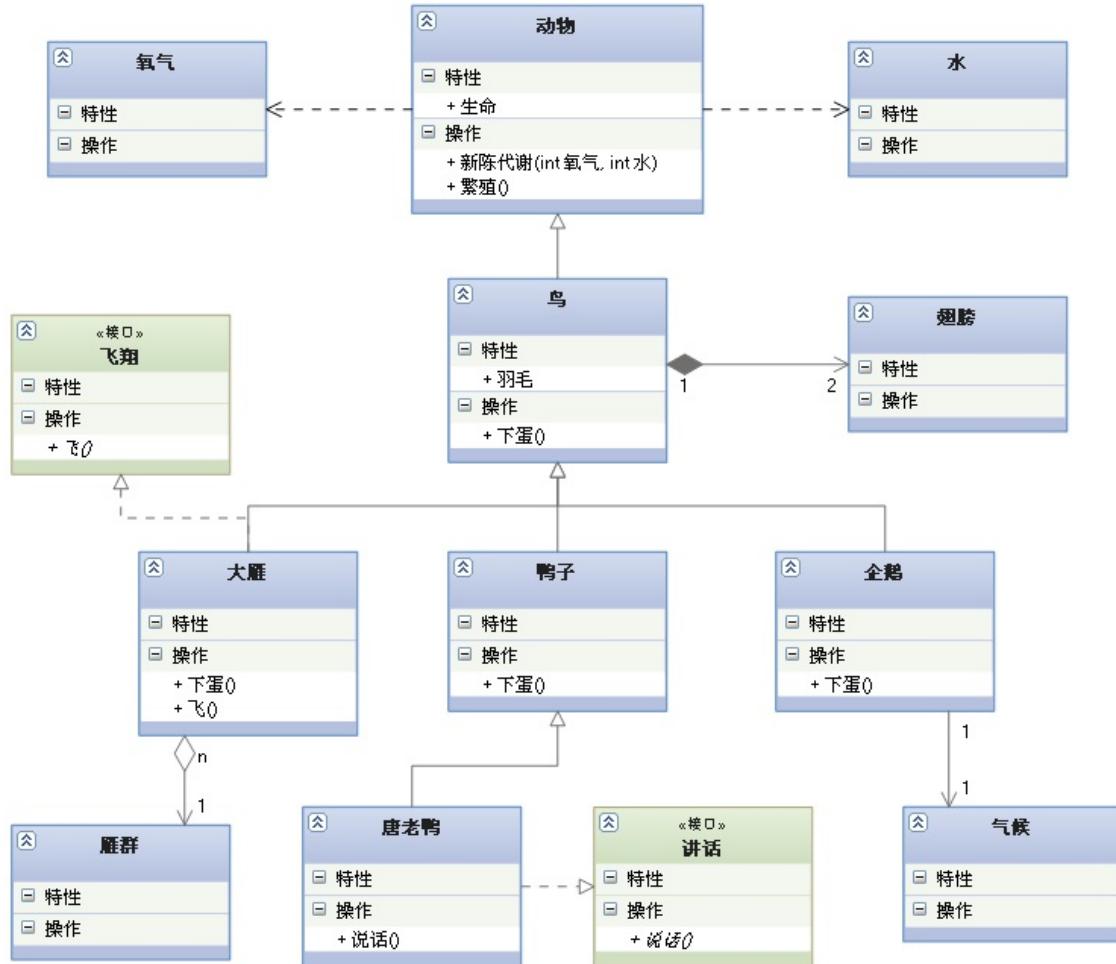
2.6. 依赖 **【依赖关系】**：是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖。

【代码表现】：局部变量、方法的参数或者对静态方法的调用



【箭头及指向】：带箭头的虚线，指向被使用者

2.7 各种类图关系



3、对象图 (Object Diagram) :

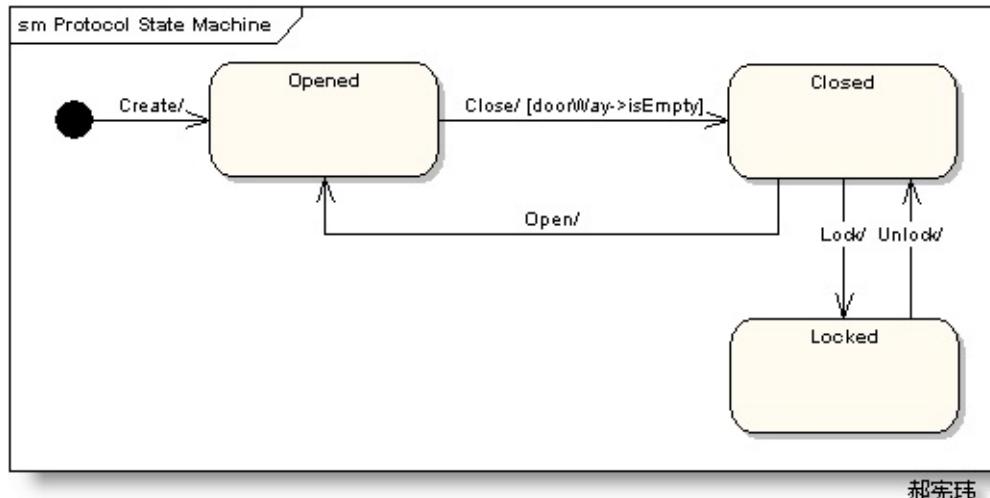
描述的是参与交互的各个对象在交互过程中某一时刻的状态。对象图可以被看作是类图在某一时刻的实例。



图 6-27 对象图

4、状态图 (Statechart Diagrams) :

是一种由状态、变迁、事件和活动组成的状态机，用来描述类的对象所有可能的状态以及时间发生时状态的转移条件。



郝宪玮

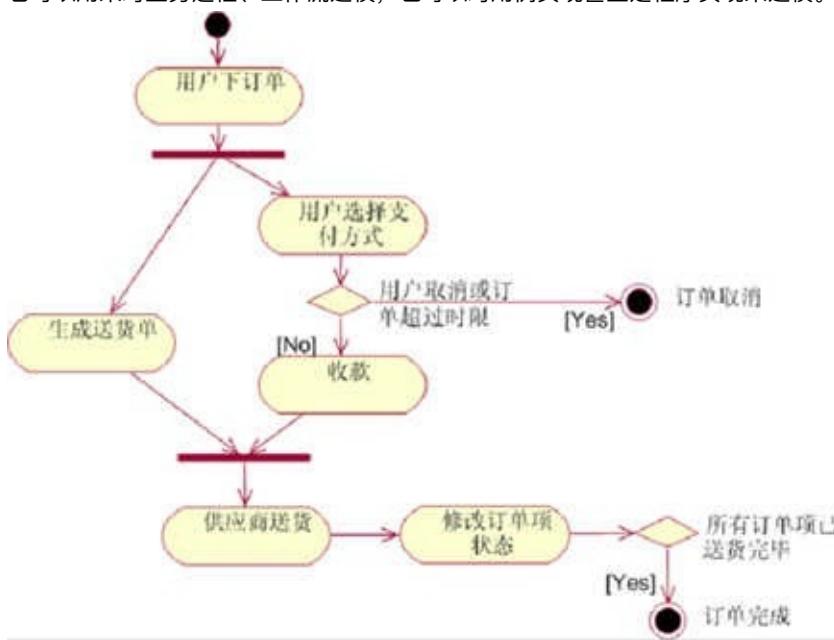
5、活动图 (Activity Diagrams) :

是状态图的一种特殊情况，这些状态大都处于活动状态。本质是一种流程图，它描述了活动到活动的控制流。

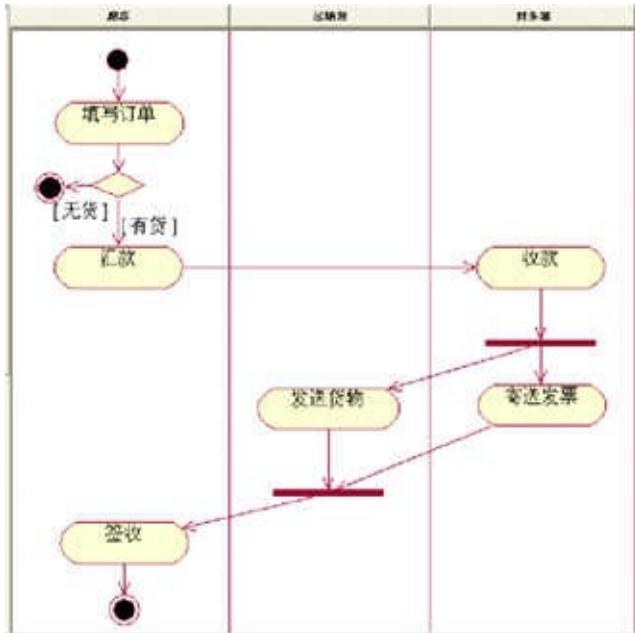
交互图强调的是对象到对象的控制流，而活动图则强调的是从活动到活动的控制流。

活动图是一种表述过程基理、业务过程以及工作流的技术。

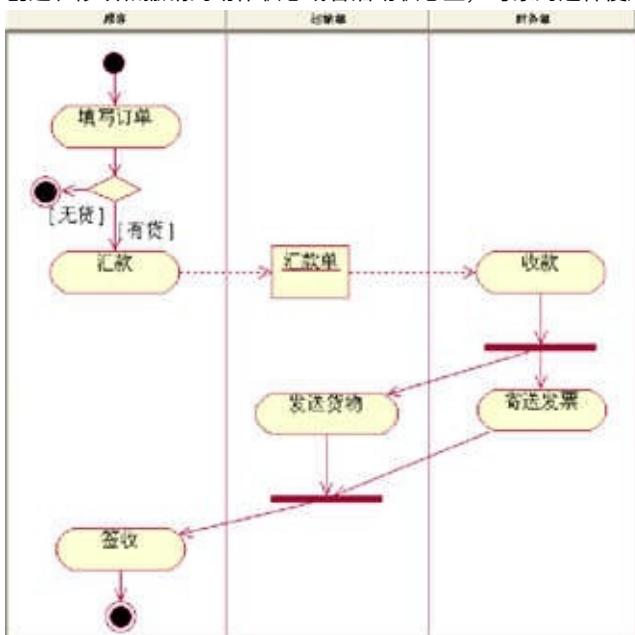
它可以用来对业务过程、工作流建模，也可以对用例实现甚至是程序实现来建模。



5.1 带泳道的活动图 泳道表明每个活动是由哪些人或哪些部门负责完成。



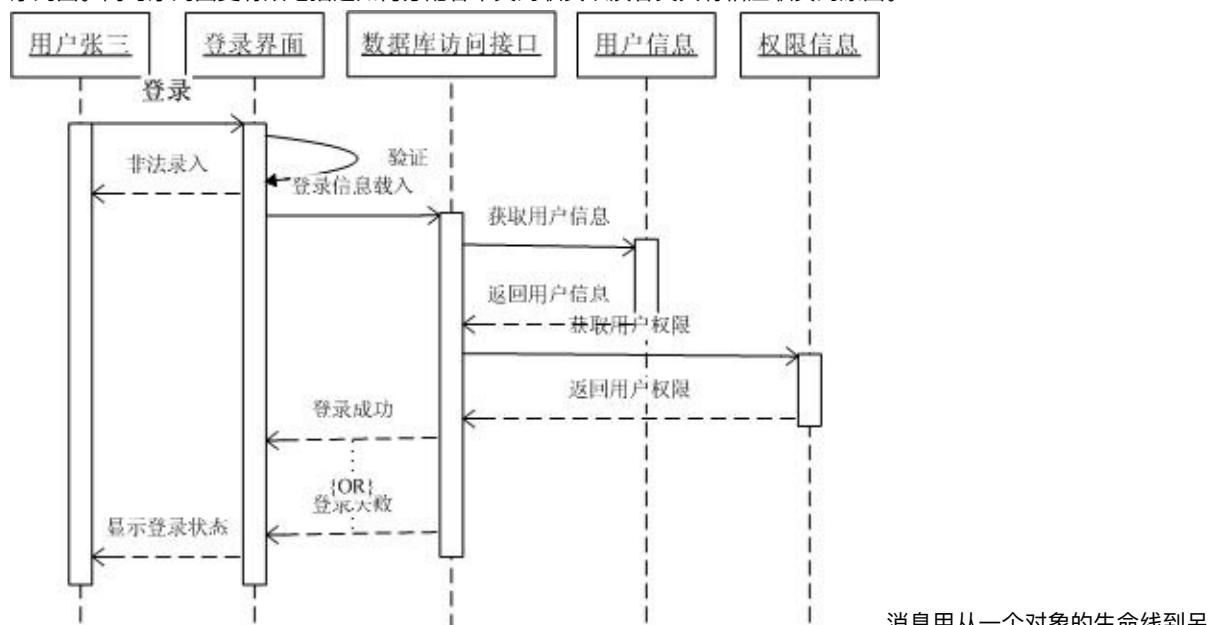
5.2 带对象流的活动图 用活动图描述某个对象时，可以把涉及到的对象放置在活动图中，并用一个依赖将其连接到进行创建、修改和撤销的动作状态或者活动状态上，对象的这种使用方法就构成了对象流。对象流用带有箭头的虚线表示。



6、序列图-时序图（Sequence Diagrams）：

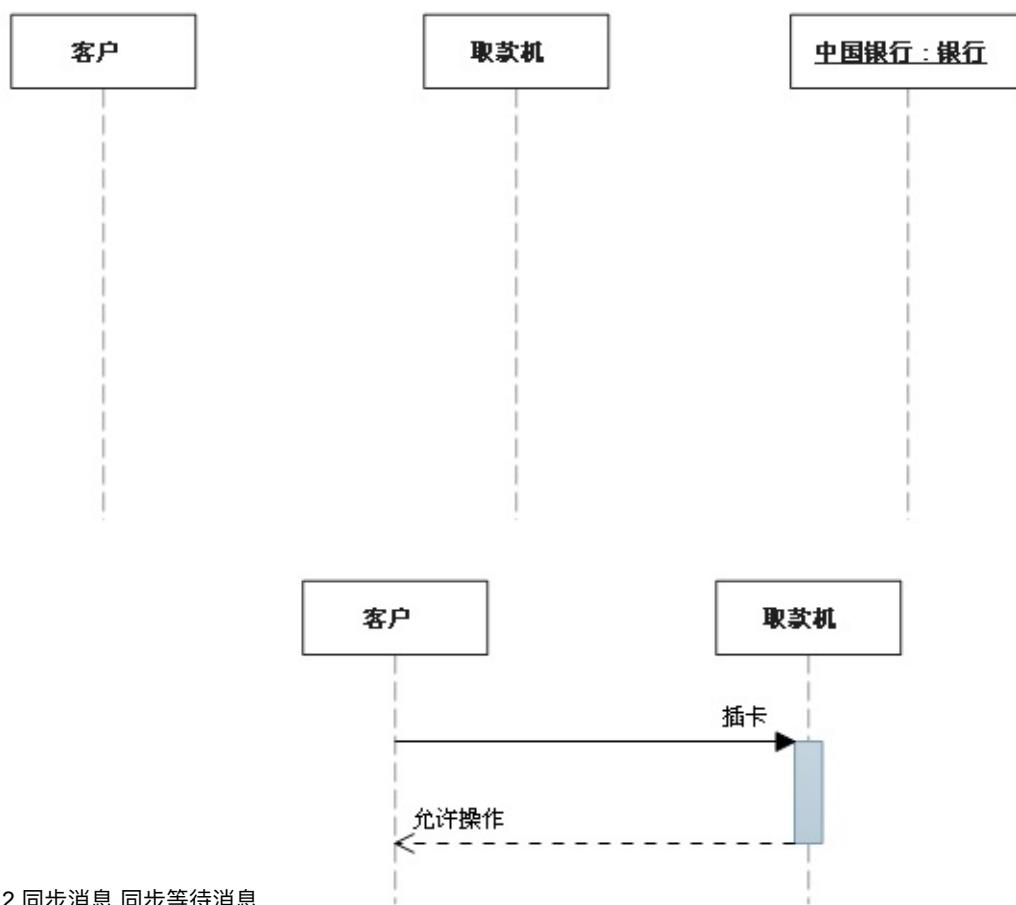
交互图的一种，描述了对象之间消息发送的先后顺序，强调时间顺序。

序列图的主要用途是把用例表达的需求，转化为进一步、更加正式层次的精细表达。用例常常被细化为一个或者更多的序列图。同时序列图更有效地描述如何分配各个类的职责以及各类具有相应职责的原因。

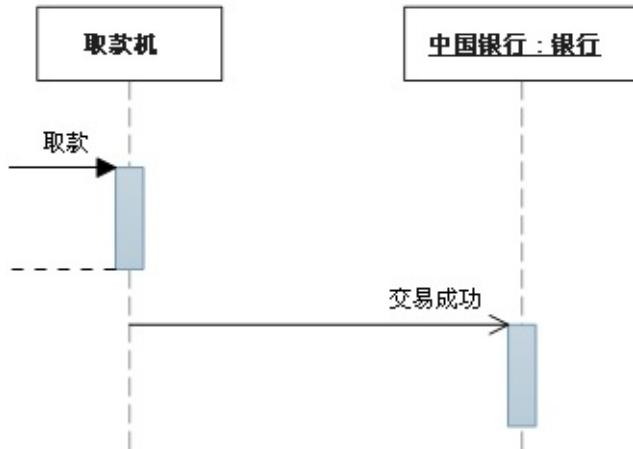


序列图中涉及的元素：

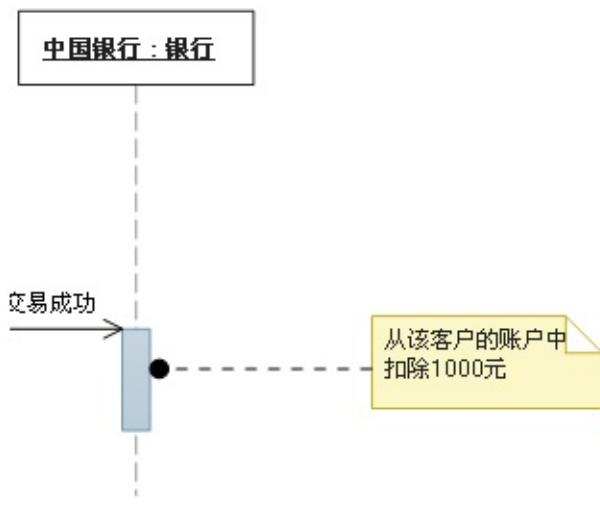
6.1 生命线 生命线名称可带下划线。当使用下划线时，意味着序列图中的生命线代表一个类的特定实例。



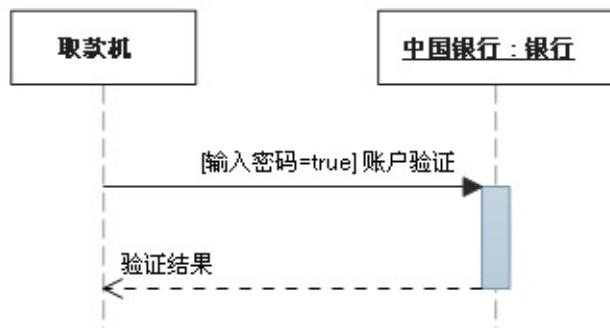
6.2 同步消息 同步等待消息



6.3 异步消息 异步发送消息，不需等待



6.4 注释

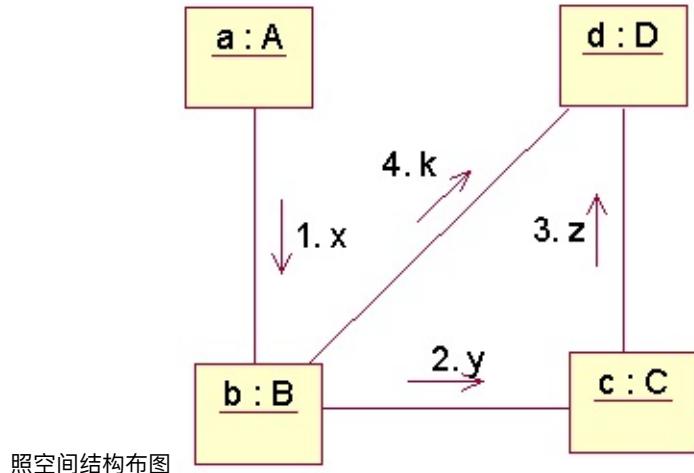


6.5 约束

6.6 组合 组合片段用来解决交互执行的条件及方式。它允许在序列图中直接表示逻辑组件，用于通过指定条件或子进程的应用区域，为任何生命线的任何部分定义特殊条件和子进程。常用的组合片段有：抉择、选项、循环、并行。

7、协作图 (Collaboration Diagrams) :

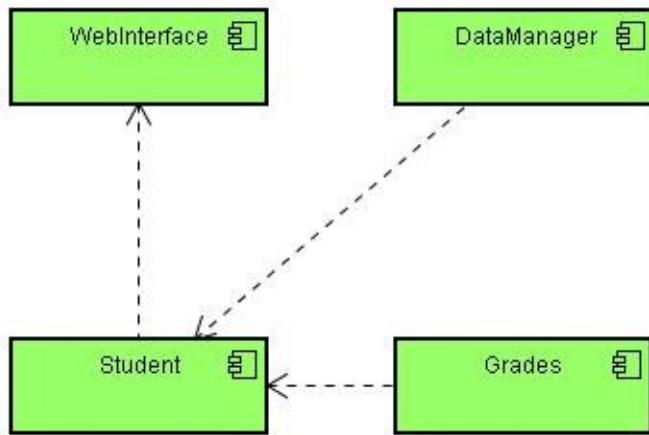
交互图的一种，描述了收发消息的对象的组织关系，强调对象之间的合作关系。时序图按照时间顺序布图，而写作图按照空间结构布图



照空间结构布图

8、构件图 (Component Diagrams) :

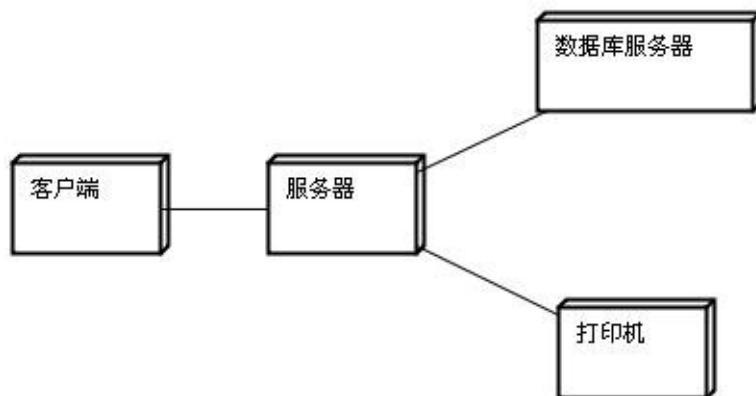
构件图是用来表示系统中构件与构件之间，类或接口与构件之间的关系图。其中，构件图之间的关系表现为依赖关系，定义的类或接口与类之间的关系表现为依赖关系或实现关系。



9、部署图 (Deployment Diagrams) :

描述了系统运行时进行处理的结点以及在结点上活动的构件的配置。强调了物理设备以及之间的连接关系。

部署模型的目的：描述一个具体应用的主要部署结构，通过对各种硬件，在硬件中的软件以及各种连接协议的显示，可以很好的描述系统是如何部署的；平衡系统运行时的计算资源分布；可以通过连接描述组织的硬件网络结构或者是嵌入式系统等具有多种硬件和软件相关的系统运行模型。



二、图的差异比较

1.序列图(时序图)VS协作图

序列图和协作图都是交互图。二者在语义上等价，可以相互转化。但是侧重点不同：序列图侧重时间顺序，协作图侧重对象间的关系。

共同点：时序图与协作图均显示了对象间的交互。

不同点：时序图强调交互的时间次序。

协作图强调交互的空间结构。

2.状态图VS活动图

状态图和活动图都是行为图。状态图侧重从行为的结果来描述，活动图侧重从行为的动作来描述。状态图描述了一个具体对象的可能状态以及他们之间的转换。在实际的项目中，活动图并不是必须的，需要满足以下条件：1、出现并行过程&行为；2、描述算法；3、跨越多个用例的活动图。

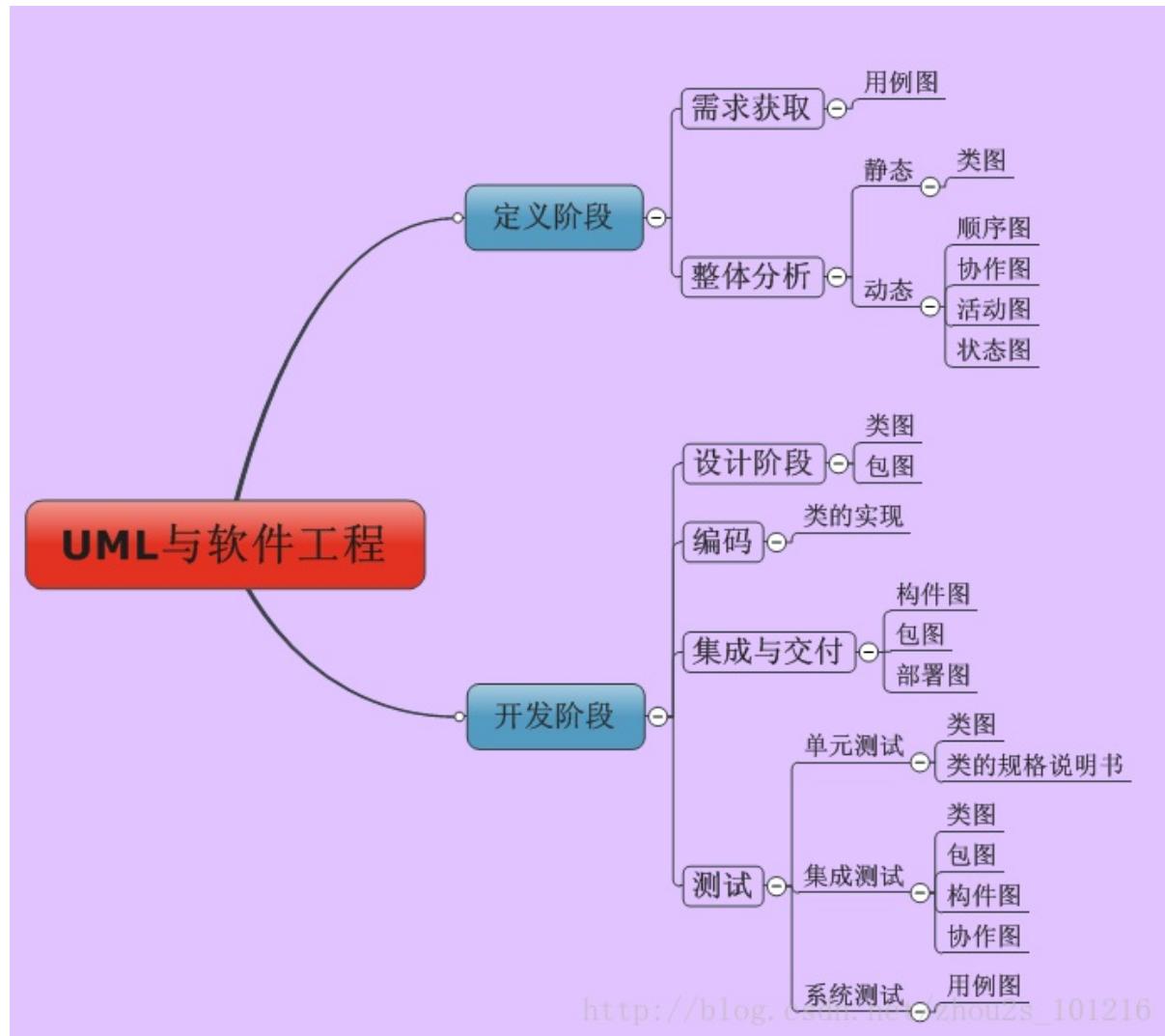
3.活动图VS交互图

二者都涉及到对象和他们之间传递的关系。区别在于交互图观察的是传送消息的对象，而活动图观察的是对象之间传递的消息。看似语义相同，但是他们是从不同的角度来观察整个系统的。

三、UML与软件工程

UML图是软件工程的组成部分，软件工程从宏观的角度保证了软件开发的各个过程的质量。而UML作为一种建模语言，更加有效的实现了软件工程的要求。

如下图，在软件的各个开发阶段需要的UML图。



下表是UML使用人员图示

UML图的使用人员					
人员	系统用户	分析人员	设计人员	开发人员	测试人员
用例图					
类图					
对象图					
序列图					
协作图					
状态图					
活动图					
构件图					
部署图					

- 类图
- 时序图

- 用例图