

软件工程

大连理工大学软件学院



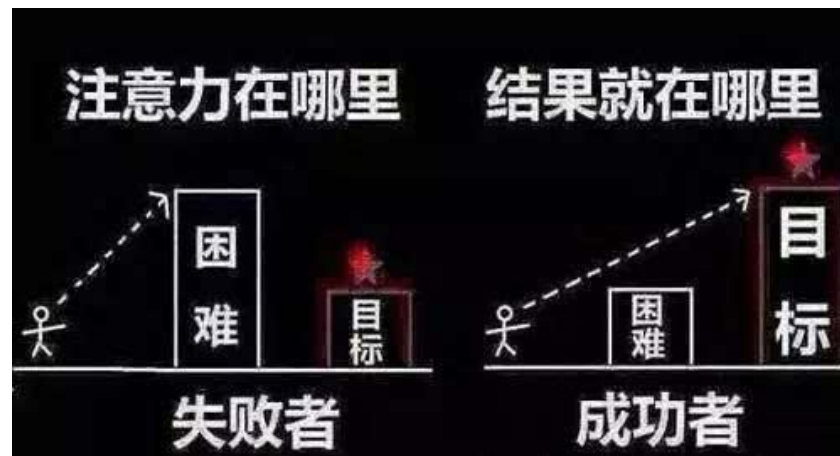
第3章 需求分析

- 需求分析的目标就是搞清楚用户真正想要的系统是什么以及存在哪些约束条件。
- 需求分析是软件开发的输入，“垃圾入，垃圾出”，所谓“失之毫厘，谬以千里”。
- 需求分析的捕获和描述、需求分析的细化、功能性需求和非功能性需求等。



需求分析的挑战

- 无章法的软件开发，尤其是在开发过程上游中的需求分析，很容易导致整个项目的失败
- 分析阶段开发人员需要与未来系统相关的领域专家们在一起工作。对于软件开发者来说，这是一个学习未知领域的过程，是建造一个新的世界的开始



需求分析的挑战

- 分析人员借助各种手段熟悉、了解和掌握相关的业务领域，这不同于简单的记录，对于业务的理解一定要经过大脑的思考和整理，这样的解决方案才会奏效
- 每个人包括领域专家们对于系统的全局都是一个局部，很难预知哪些领域信息会对以后的开发产生影响
- 客户的想法会在开发进行中毫无征兆的发生改变，如何采取措施来应对这些无法预知的变化

需求分析的开始

- 需求分析的开始阶段，往往还伴随着一个相对较短的可行性分析活动，给出系统的一个合理可行的解决方案。
- 可行性分析的结果还有另外一个作用，就是评估出系统初步的开发费用，这也是与客户（甲方）签订商务合同的依据。
- 初步的需求分析也可由项目的委托方组织相关的技术人员进行实施，并给出系统主要的需求列表。
 - 由客户为主导制定的需求文档称为用户（业务）需求
 - 由开发者为主导制定的需求文档称为系统需求

用户需求与系统需求

- 系统需求是对用户需求的细化和完善。
- 系统需求的阅读对象是开发者，而用户需求的阅读对象是委托方或客户。
- 系统需求是用户需求的开始，用户需求需要得到委托方的确认。
- 无论是系统需求还是用户需求，在调研的时候都需要搞清楚以下两个基本问题：
 - 项目涉及到的主要目标人群有哪些？
 - 项目主要的目标是什么？

涉众（Stakeholder）

- 是与目标系统相关的一切人和物，他们对目标系统的构建会有一定的影响。
- 是一个比较宽泛的概念，可以是提出系统原始需求的人，可以是委托方，也可能是使用目标系统的最终用户等等。
- 涉众可能并不固定，有时会对系统的设计策略和实现方式有着深远的影响。
- 涉众的重要程度是不同的。
 - 准确的识别出来，并确定其优先级
 - 需要斡旋和协调，让目标系统满足大多数涉众的要求

常见的涉众

- 最终用户：系统的实际使用者，对目标系统有直接的接触和评价。（直接涉众）
- 投资者：或称为出资方，主要关心目标系统的总成本、建设周期以及未来的收益等高层目标。
 - 虽不直接参与系统的开发和技术，但会对实际采用的技术及边界产生影响
- 业务提出者：为业务规范和质量提升提出项目开发纲领性的指导。
 - 比较原则化和粗略化，但需要重视和贯彻
 - 通常是业务方的高层人员，如总经理，分管生产、财务、销售等部门经理等

常见的涉众

- 业务管理者：负责业务计划、生产、监督等环节的实际实施和控制。
 - 是需求分析过程中重要的调研对象，他们的需求更加面向具体业务，更为实际和现实
 - 他们对业务流程、业务规则以及业务模式等有着比较深入的理解
 - 调研过程需要科学的设计，需要引导
 - 通常是业务方的中层管理者，如各科室的主任等
- 业务的执行者：操作人员，是频繁与未来系统直接交互的人员。
 - 更为细节的业务要求，可能不合理，但容易说服
- 其他涉众：第三方、开发方、法律法规等

系统目标

- 系统的目标和涉众的确定是紧密相连的，因为目标主要决定了涉众存在的缘由。
- 要尽量将目标明确的定义下来，并同时说明对这些目标的验证标准。
- 确保目标或目标的内部之间不会有矛盾的地方，整体上项目的各个部分都要保持一致。
- 未来系统开发的纲领，用户需求应与目标进行关联。

系统目标定义模板

目标	设置该目标可以满足的期望
对涉众的影响	对应的涉众及其影响
边界条件	附加条件或者约束
依赖	是是否依赖其他目标？与其他目标的关系如何？是相辅相成还是此消彼长？
其它	其它需要说明的内容。

确定系统功能

- 根据项目类型和涉众的IT经验和技能，可采用不同的方法识别系统功能：
 - 通过座谈的方式确定未来软件支持的业务工作流程。
 - 访谈关键涉众，因为未来的系统最终由这些人员组织验收。
 - 采用调查问卷的形式，但是要求问卷中的题目要有代表性。
 - 分析旧系统或当前系统中的问题或值得借鉴的地方，挖掘优化的潜力。
 - 现场与最终用户的调研和访谈，使得最终用户逐渐融入开发过程，也能使其容易理解业务的实现原理。

领域分析、了解
背景、学习术语

- 领域专家
- 领域书籍
- 现有软件及其文档
- 其它任何材料，如网站等

观察

(简单)用
例分析

访谈

正式

非正式(休闲聊天、
面对面)

规格说明
(头脑风暴)

原型化

- 访谈有两种基本形式，分别是**正式**的和**非正式**的访谈。

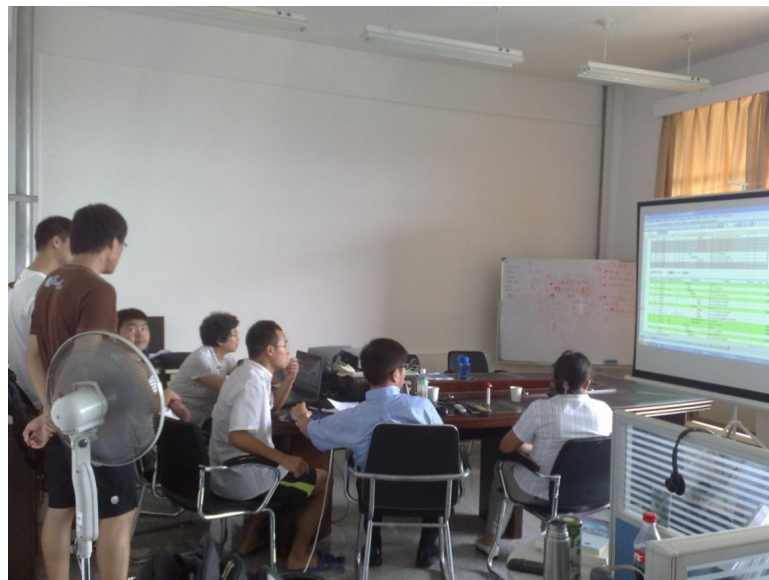
- 正式：系统分析员将提出一些事先准备好的具体问题。
- 如，询问客户公司销售的商品种类、雇用的销售人员数目以及信息反馈时间应该多快等。



- 在非正式的访谈中，将提出一些可以自由回答的开放性问题，以鼓励被访问的人员表达自己的想法，例如，询问用户为什么对目前正在使用的系统感到不满意。

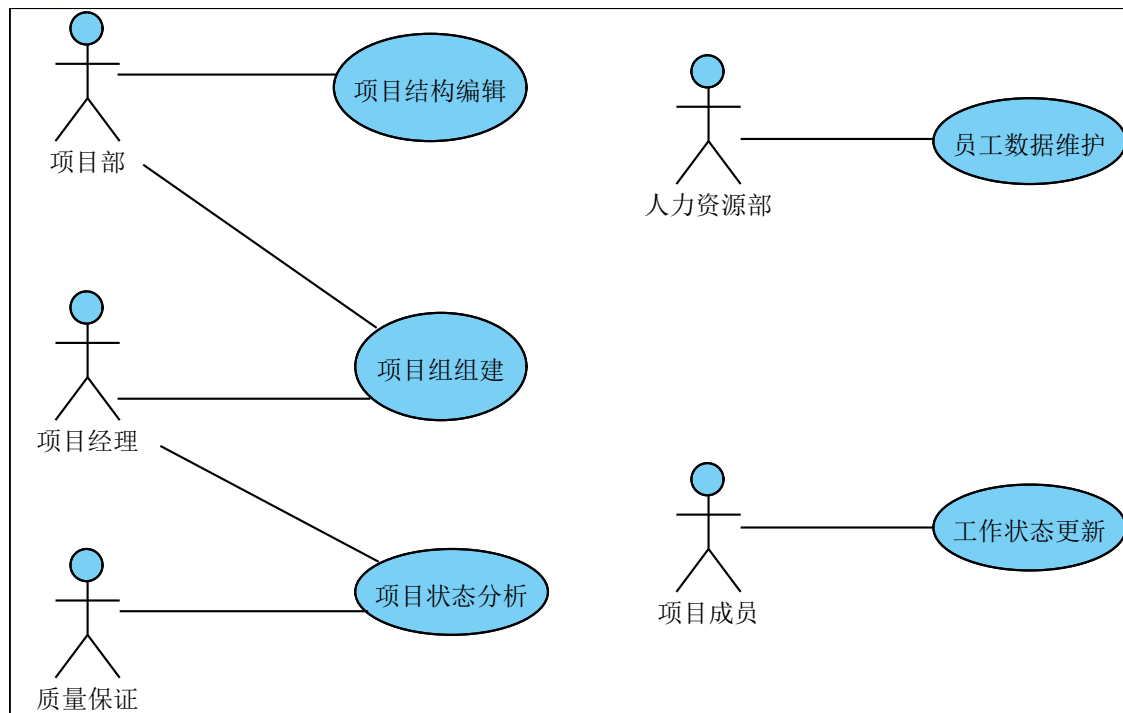
用例（Use case）

- 使用一种交互的方式来描述系统的**场景**，借以“捕获”用户的需求。
- “捕获”指对业务用例不是简单静态说明，要构建动态的场景，强调的参与者的活动。
- 用例又称为是用户故事（User Story），是对需求的深入分析和理解的输出结果。
 - 用例的完善需要迭代，每次添加一些业务细节



- 用例规约，对具体用例场景中业务流程的脚本式的说明。

用例的表示

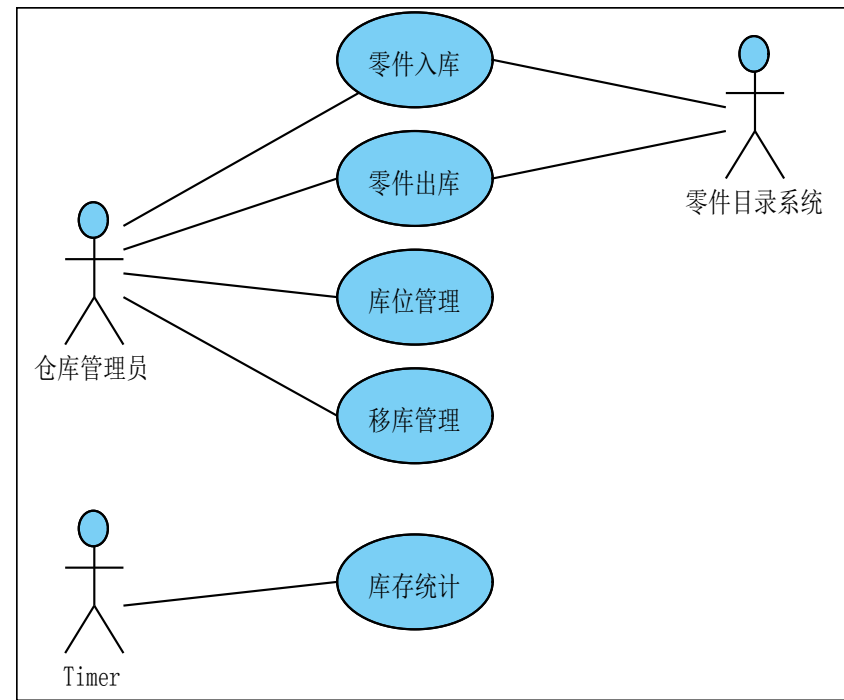


- 用例使用椭圆表示，代表用户任务
- 任务对应的涉众，用例直接或间接与人形符号（Stickman）关联，叫做Actor。
- Actor理解成角色（Role）会更合适一些，指使用系统的用户类，不特指具体人。


- 不同的人可以扮演相同的角色，相同的人也可能在不同的场合中扮演不同的角色，角色与具体的人或职位并没有必然的对应。
- 角色实现系统的某些用例，一个角色可以对应多个用例；相反，一个用例也可以对应多个角色的参与。

识别角色

- 在实际应用中，角色可以帮助来寻找用例
- 角色是存在于系统边界之外的与系统发生某些交互的对象
- 角色可以是人，也可以是其它软件系统，如“零件目录系统”和“定时器（Timer）”
- 与“定时器”关联的用例表示该用例是在特定的时间被自动触发，产生对该仓库的统计报告



寻找用例

- 
1. 考虑在业务系统中进行管理和处理的关键业务实体，通常是与业务相关的一些关键概念或技术术语，如：
 - 软件项目（Project）：每个软件项目对应着一个在系统中组织和管理的基本结构，项目开发者以团队的形式参与其中。
 - 员工（Employee）：即开发者，是项目实施的基本单位，可进一步组成团队的结构。

每个业务实体对应5个基本用例（根据情况合并、正名）

1. 创建新业务实体数据的用例；
2. 修改已有业务实体数据的用例；
3. 删除已有业务实体数据的用例；
4. 持久化某业务实体数据的用例，比如存储到数据库中；
5. 访问某业务实体数据的用例，比如从存储的文件或数据库中读出。

寻找用例


2. 考虑业务相关的过程数据，即围绕基本业务实体的加工和组合而产生的数据，会在业务执行期间频繁和显著的发生变化。

- 项目组：对员工进行组织，并分配到具体的项目中。项目组的各种数据在项目进行过程中会经常发生改变；
- 工作时间：任务的起始和结束时间，开发者需要按照任务的时间安排开展工作。

可按照1中的方法确定用例：

- 如“项目团队组建”和“项目安排更新”
- 是系统的核心用例，描述系统的主要功能需求，往往在访谈过程中由用户直接提出

寻找用例

- 
3. 进一步研究系统功能，这些功能往往需要在以上识别出的基本实体数据和动态过程数据的基础上做进一步的利用和计算。
 - 对各种数据的组合、分析和挖掘等。
 - 每个这样的数据利用，都会导致新的用例的产生。
 - “项目状态分析”用例需要将进行中的项目数据做严格的分析和评估，并可能形成项目延期的结论。
 4. 需要考虑是否存在对正在运行的其它系统的交互。
 - 在分布式的环境中，与其它系统的交互，如启动、停止或者监听数据等场景都要设置单独的用例来捕获其功能需求。

用户用例和系统用例

- 业务用例：是从客户的角度出发描述某个业务的具体 workflow，也是一次涉众与实现业务目标功能之间的交互，其中可能包含手工和自动化的过程，也可能发生在一个长期的时间段中。
- 系统用例：是从计算机系统的角度描述业务系统，其业务边界就是这个计算机系统的设计范围。
 - 主角是系统的参与者，与计算机系统一起实现一个目标。
 - 用来描述参与者如何与计算机技术相联系以及与计算机系统交互的过程，而不是详细的业务流程描述。
- 概括的说，一个业务用例描述的是业务过程——而不是软件系统的过程，一个业务用例为涉众创造价值，业务用例可以超越系统的边界。

用例规约1

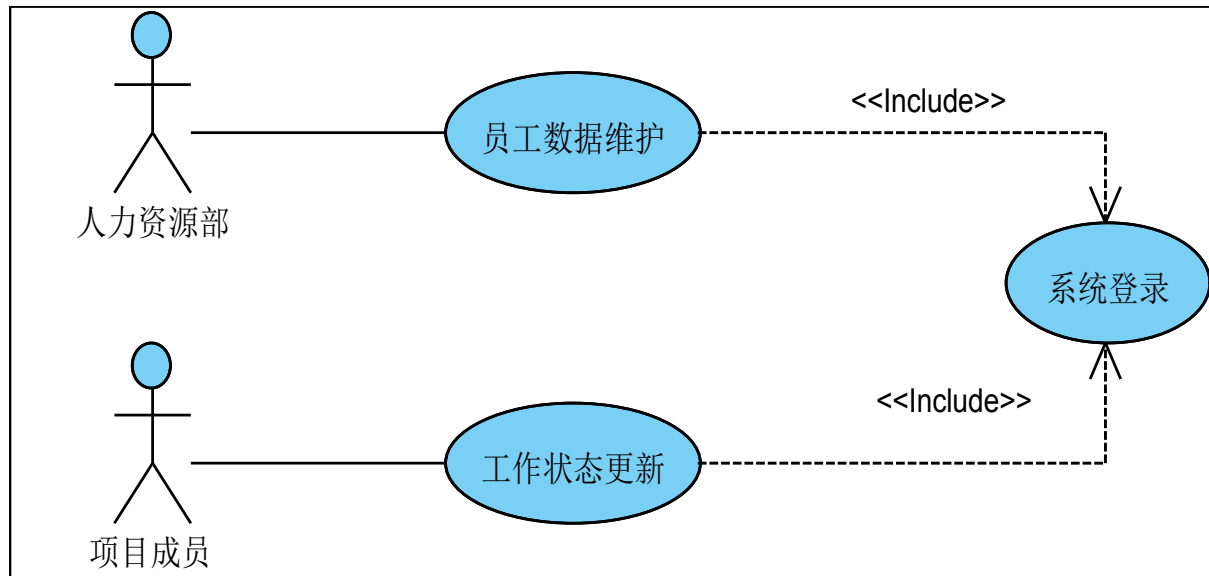
用例名称	1	简短精炼的描述，一般为动宾短语的形式。
用例编号	1	项目中唯一确定的数字编号。
包	2	在较复杂的系统中，用例被划归为不同的业务子系统，可以使用UML的包进行封装。在用例识别过程中可以确定用例归属的包。
维护者	1	创建和维护该用例的人员。
版本	1	当前用例的最新版本号，或者将版本变化的历史一同保留，记录谁在什么时间改动了哪里。
简介	1	简短描述该用例通过何种方式实现了什么功能。
参与的角色	1	参与该用例的角色（涉众），该用例对应的原始期望者。
业务支持者	1	对该用例的问题对应哪些业务人员负责解答，分别处在哪些领域。如果需要修订，谁可以决定该用例的业务内容。
引用	2	指出对该用例的实现有影响或有联系的所有信息来源，可能是某些规则、标准或现有的文档。

用例规约2

前置条件	2	执行用例之前系统必须所处的状态或达到的条件要求。
后置条件	2	用例执行完毕后系统可能处于的状态或结果。
基本事件流	2	该用例正常执行的一系列活动步骤来响应参与者提出的服务请求。
备选事件流	3	基本事件流中异常或特殊情况的处理流程。
关键性	3	该用例在系统中的重要程度。
关联用例	3	其它相关的用例。
功能需求	2	有哪些具体的功能性需求是从这个用例派生的。
非功能需求	2	有哪些具体的非功能性需求是从这个用例派生的。

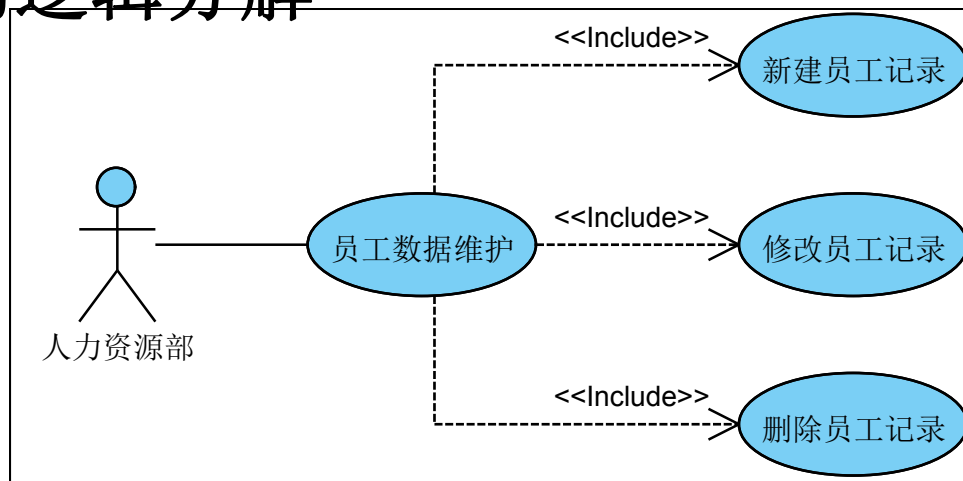
用例提炼——包含关系

- 通过一些通用过程的定义为其它主用例提供某种共同的基础性的功能
- 为避免用例中相同的基础性功能被多次重复实现
- 将这些基础功能作为一种附加用例表示，通过构造型 **<<include>>** 关系与依赖它的主用例相连



包含关系

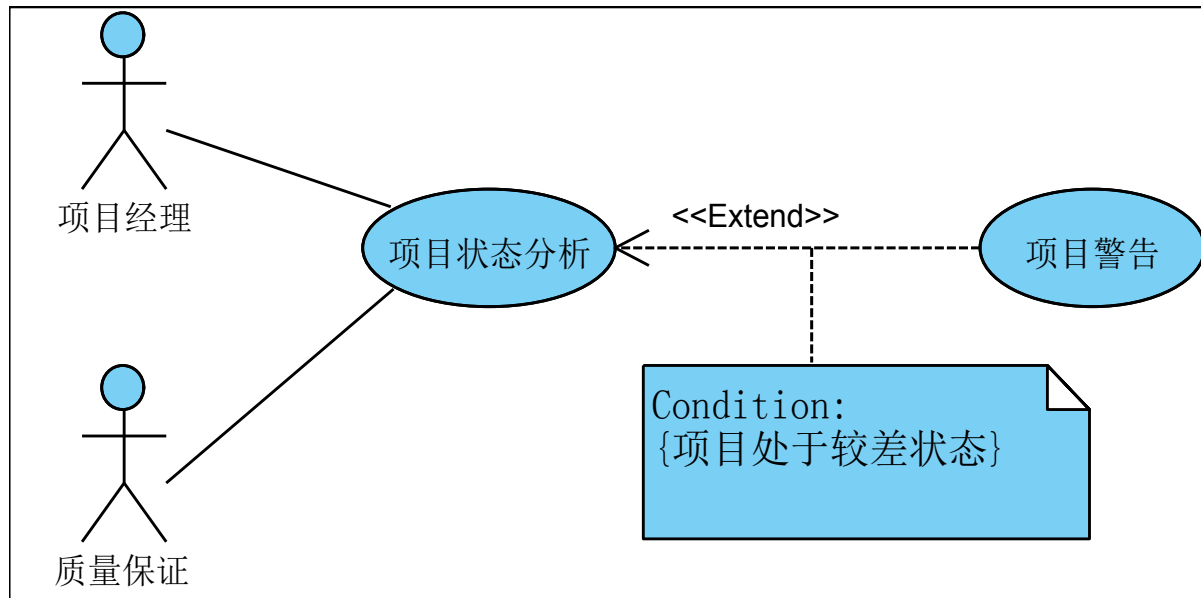
- 被包含用例也可使用构造型<<secondary>>
- 包含用例记录在用例规约中的“关联用例”字段
- 在事件流的描述中也要指明在何处被使用
- 被包含的用例应为完成具体任务的用例，避免简单的逻辑分解



错误的用例
关系

用例提炼——扩展关系

- 扩展关系<<extend>>强调客户期望中在某些情况下需要强烈表达出的一种意愿，比如特殊情况的处理



扩展关系与包含关系

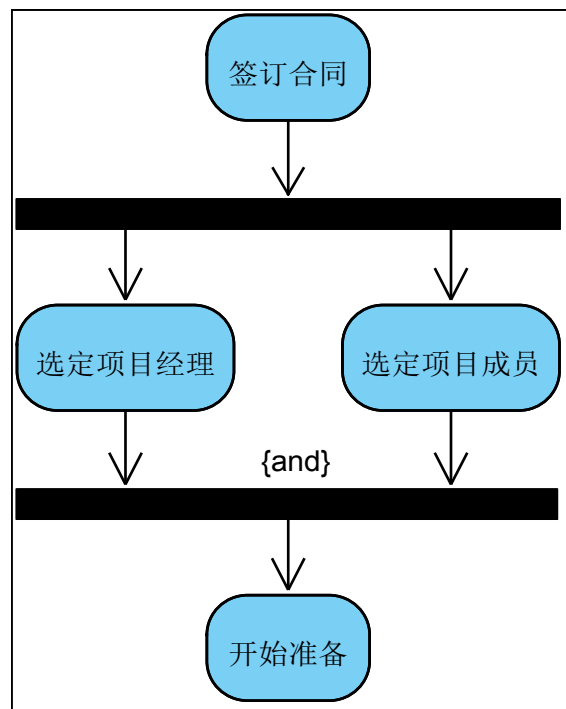
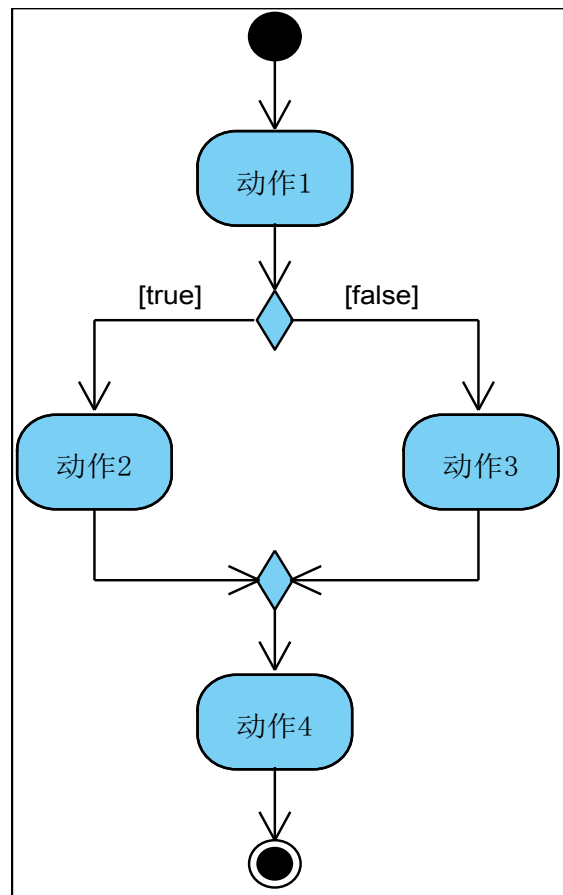
- 一般来说被包含用例属于无条件发生的用例，而扩展用例属于有条件发生的用例；
- 被包含用例提供的是间接服务，扩展用例提供的是直接服务；
- 而且扩展用例在用例规约中一般作为基本事件的备选流而存在。
- 扩展关系与包含关系要注意区分，不小心很容易混淆，需要设计人员仔细甄别正确的业务含义及其关系。

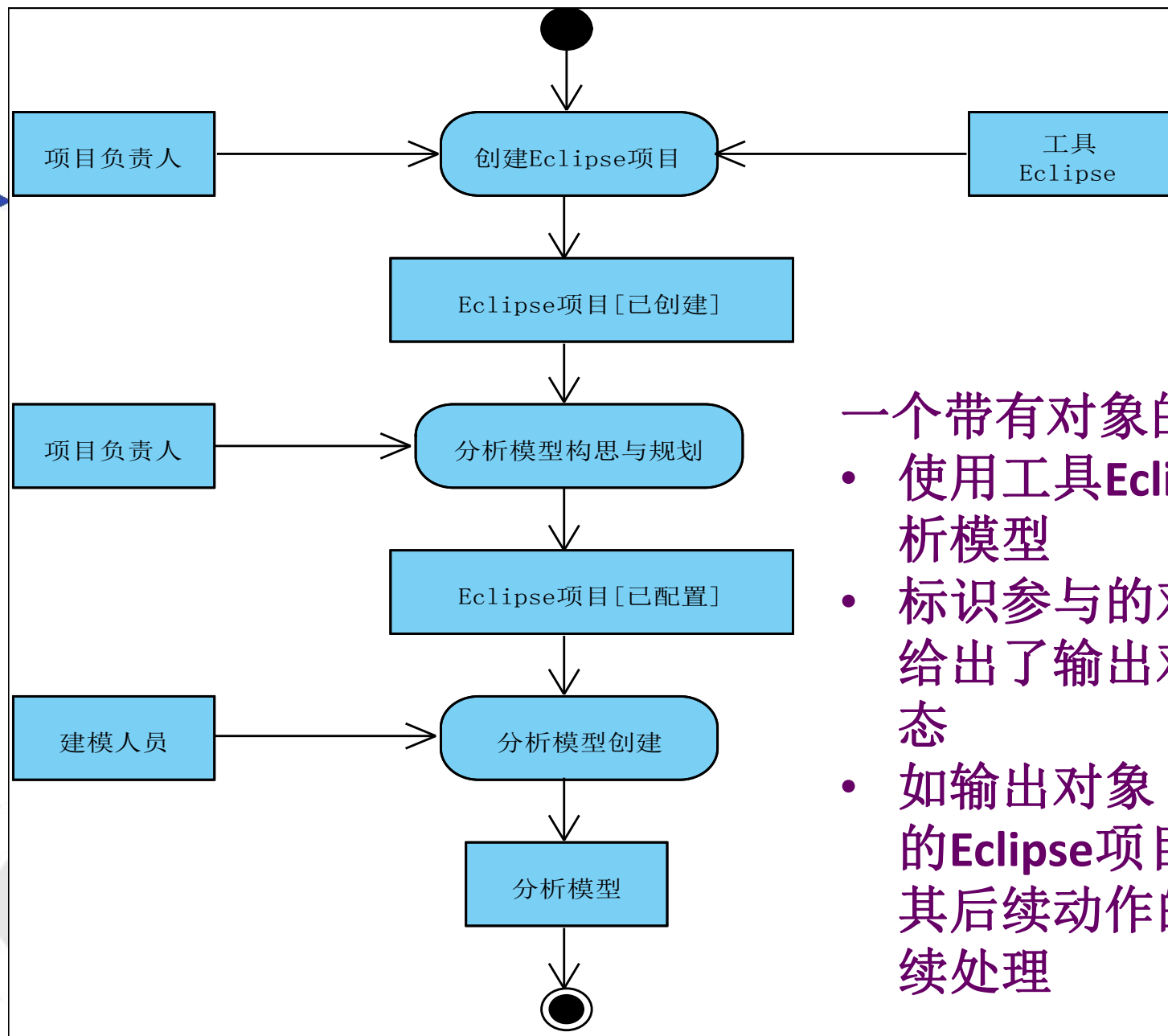
用例优化

- 即使是在非常复杂的系统中，每个用例图中的用例数目一般要避免超过15个。
- 同一用例图中的用例应尽量具有相同的业务粒度水平以及处于同一抽象级别。
- 由于在实际的设计中，上述的条件在形式上比较难以掌控，建议将所有的用例放在一起进行筛选和分级。
- 从用例出发：
 - 以此进行进一步的开发和细化；
 - 以此作为出发点，进行较详细的成本估算；
 - 作为增量或迭代计划的基础元素；
 - 用于不同团队之间的工作分工。

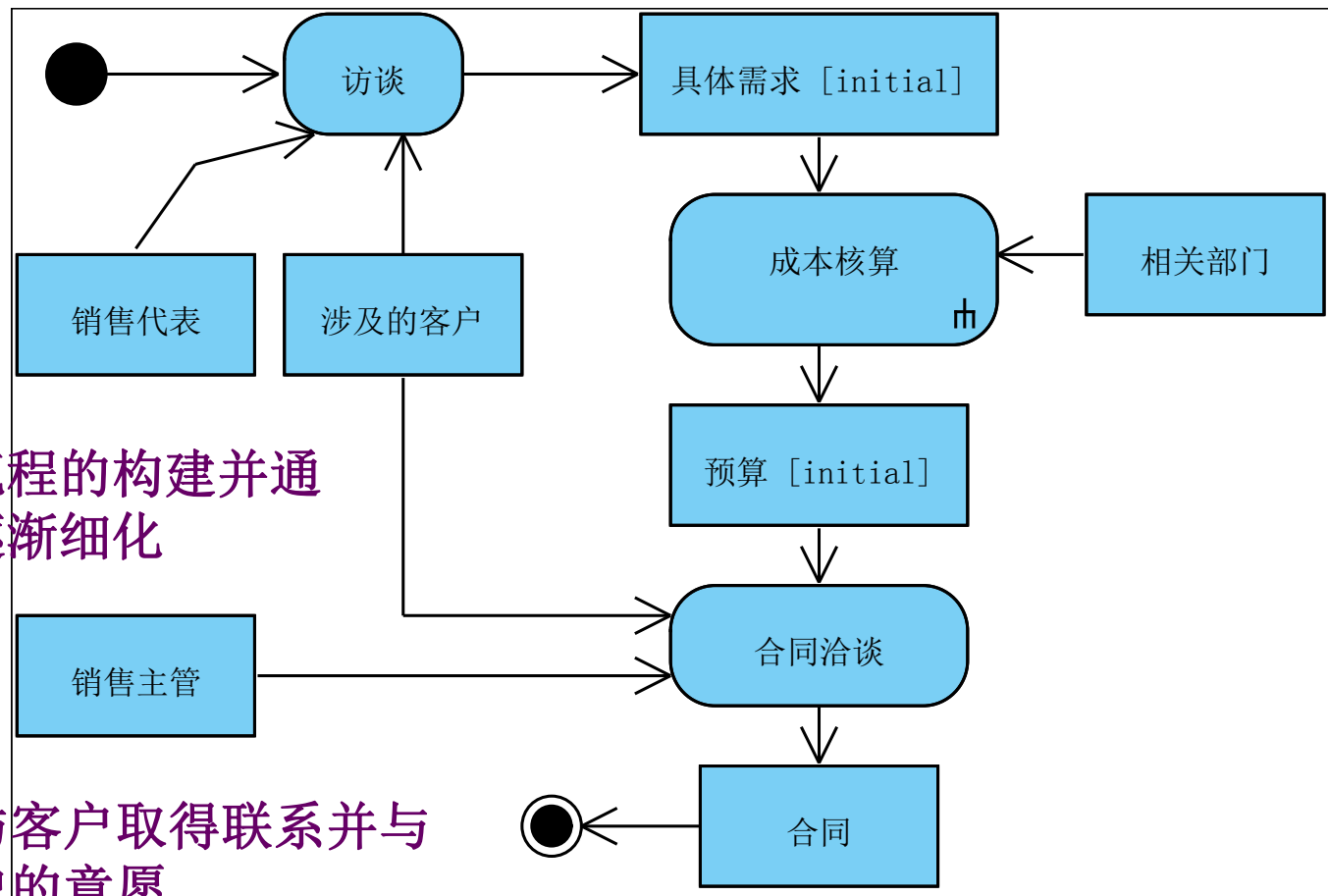
活动图进行过程建模

- 对于处理流程的建模，使用UML中的活动图是非常适合的
- 开始点、结束点
- 动作及执行顺序
- 条件
- 分支、汇聚 {and, or}
- 对象



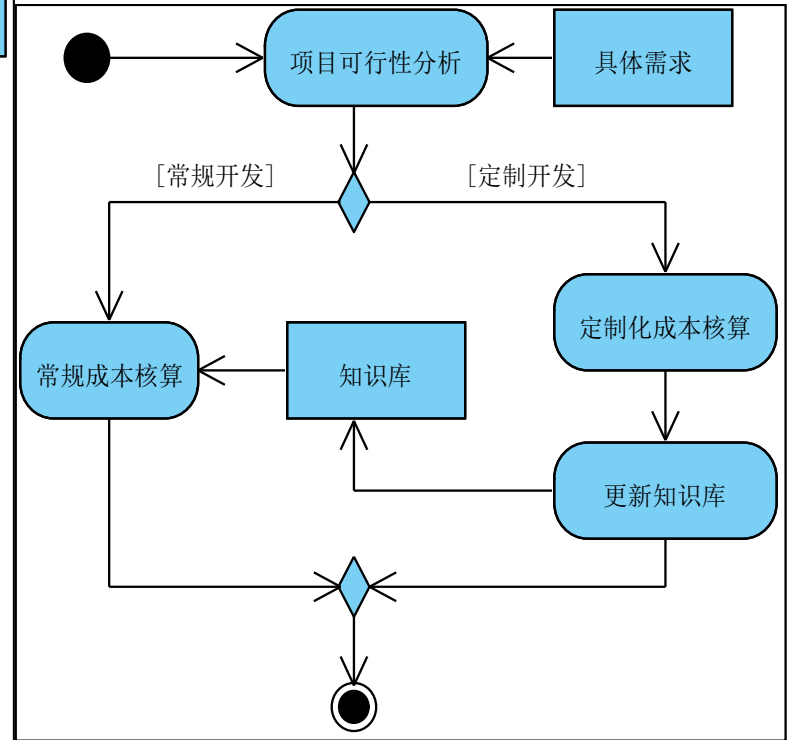
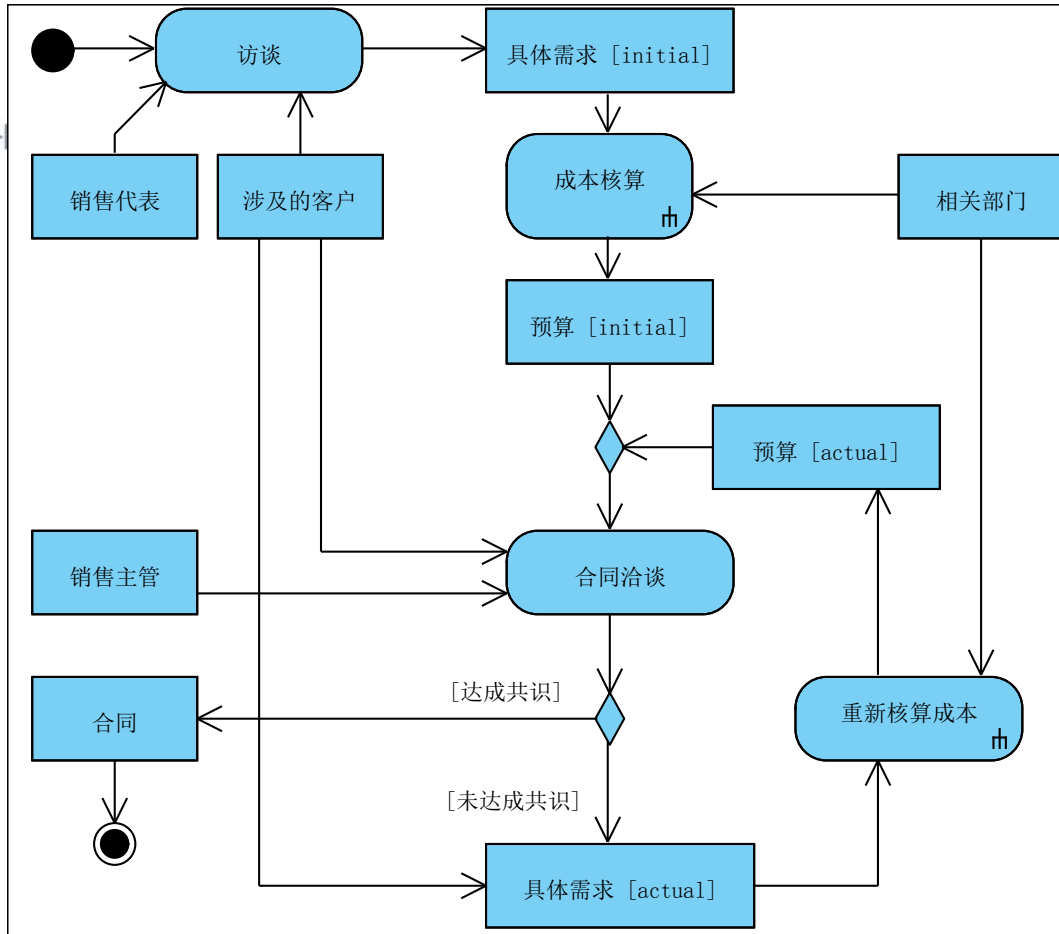


- 一个带有对象的活动图：
- 使用工具Eclipse创建分析模型
 - 标识参与的对象并且给出了输出对象的状态
 - 如输出对象“已创建的Eclipse项目”会作为其后续动作的输入继续处理

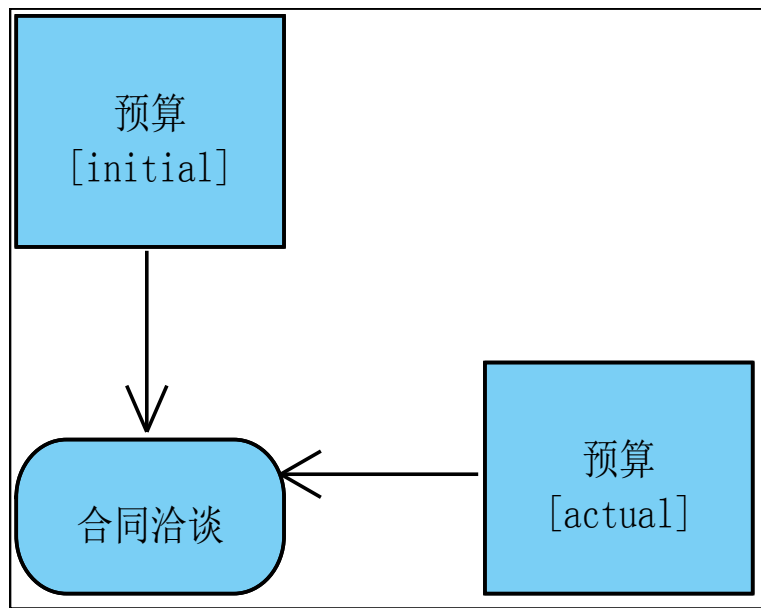


- 一个销售业务流程的构建并通过增量的方法逐渐细化
- 销售代表首先与客户取得联系并与其沟通确定客户的意愿
- 相关部门提供成本预算，然后与客户签订合同，最后是项目的实现（已省略）。
- 识别出的角色包括销售代表、客户以及可能的相关（技术）部门，产品包括客户的意愿、成本预算以及合同。

- 动作“成本核算”使用了一个特殊符号标记，其表示此动作通过另外一个活动图进行了细化



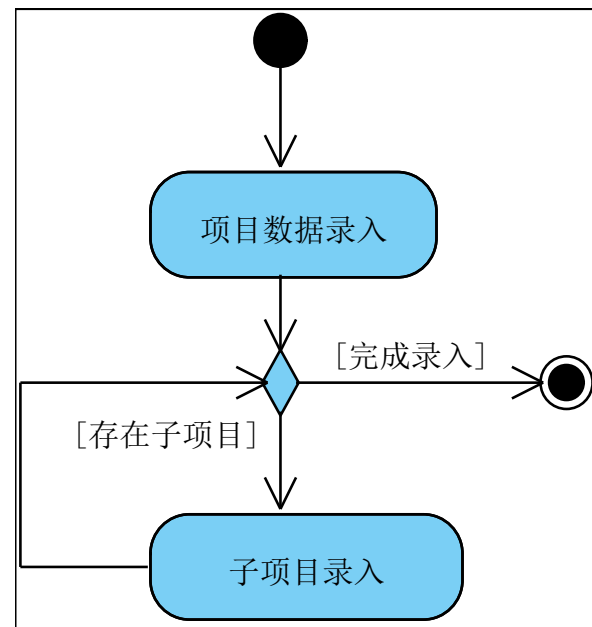
- 细化的活动图补充了一些可能的流程，去掉了参与者的角色说明，因为按照上层的活动图可知；所有活动都应由某个相关部门的员工负责



- 在合并加入新的流程时，总是伴随着要补充一个控制点的菱形
- 一种简化的形式，如图中所示，省略对应的菱形控制点
- 活动图的语法规则：如果动作具有多个汇聚的箭头，则需要等待所有的分支都完成，类似于并行分支的同步情况

基本事件流

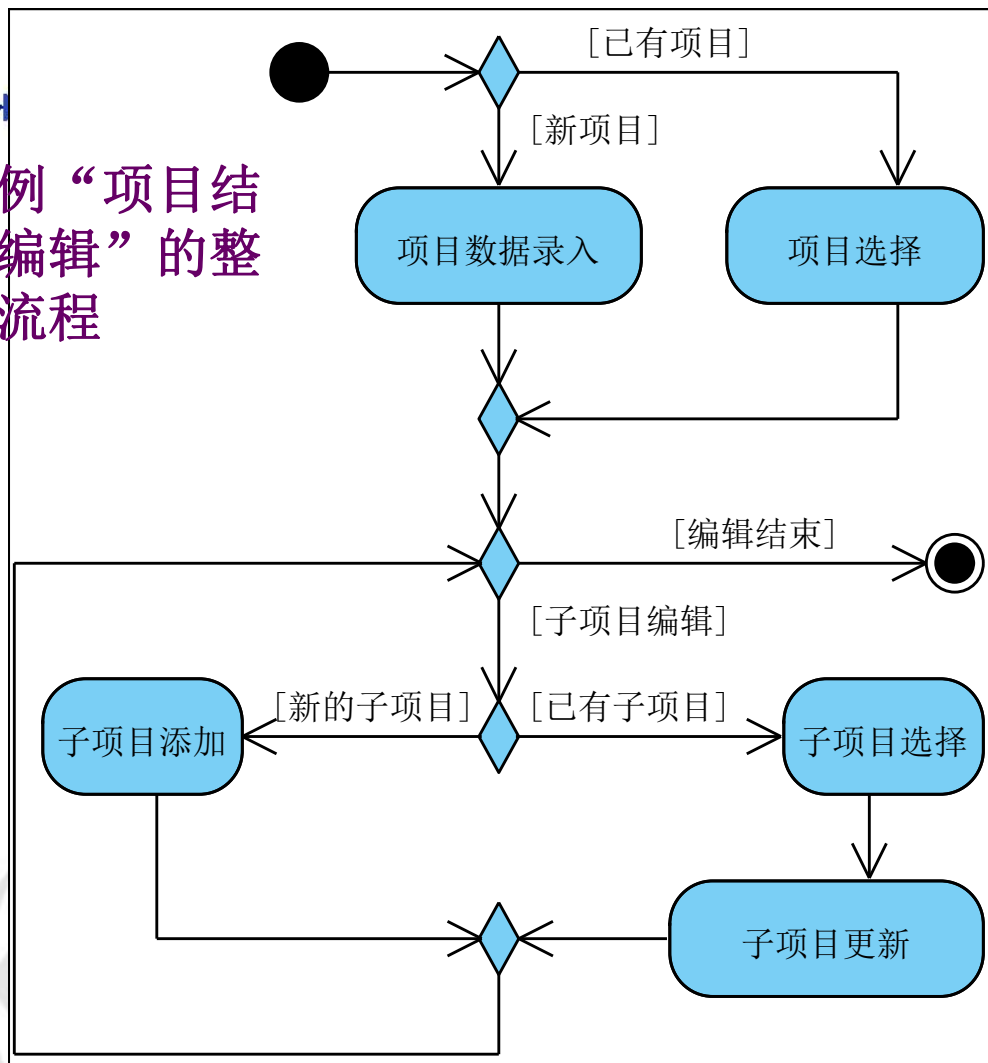
- 对于处理流程的描述活动图是非常适合的，可以使用活动图对基本事件流和备选事件流进行描述。
 - 对于用例中基本流的每一个步骤，生成一个单独的动作（action）；
 - 如果这个动作比较复杂，可以对应使用几个动作来对它进行表示，或者使用另外一个单独的活动图对其进行细化；
 - 逐步补充每个备选流的动作，同时注意每个动作是否需要进一步的细化。



用例“项目结构编辑”的典型流程

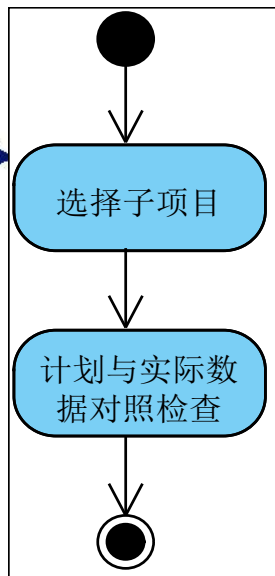
补充备选流

用例“项目结构编辑”的整体流程

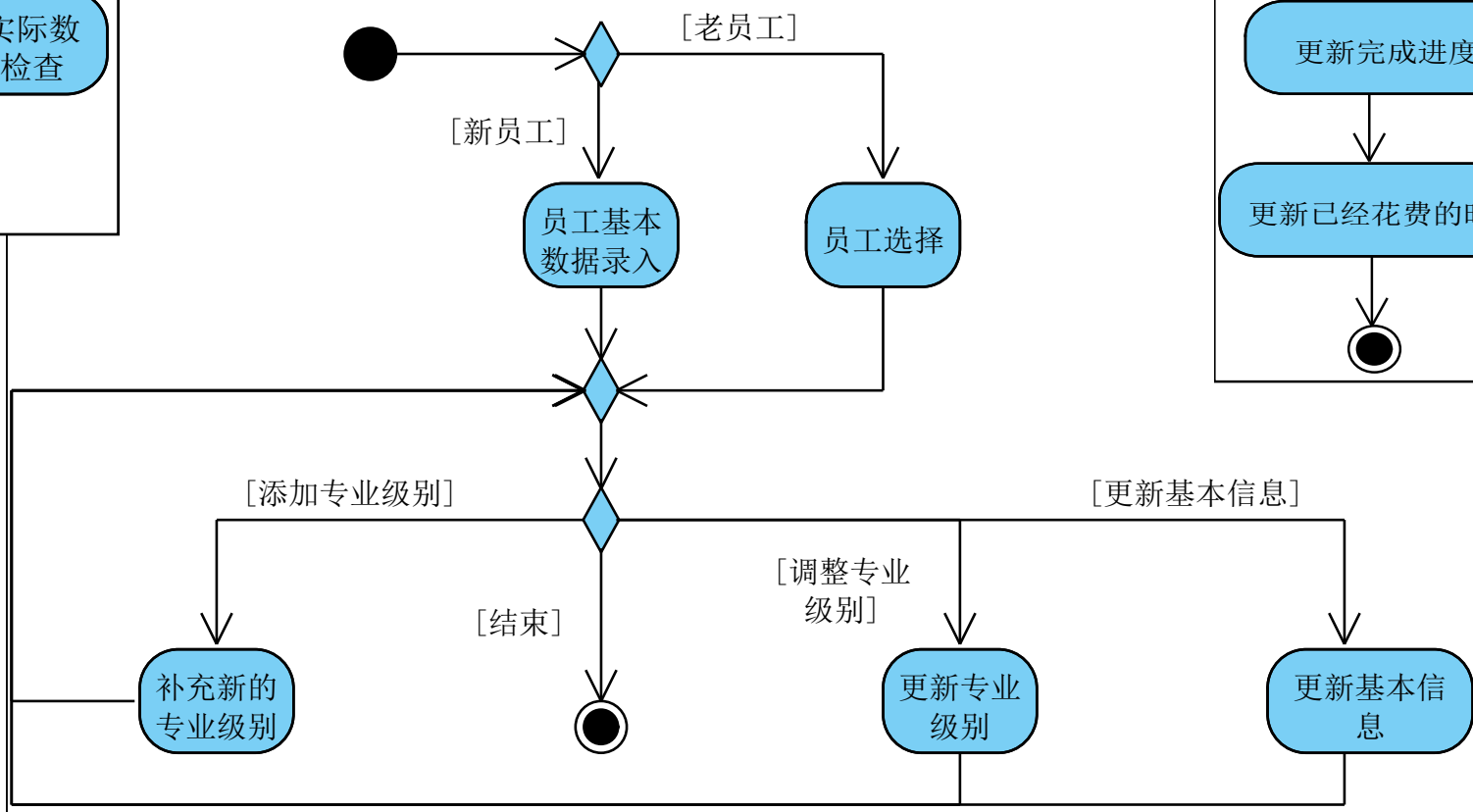


- 备选流的动作应与现有活动图进行衔接，比如“输入数据的合理性检查”有必要加入进来，并对其执行条件做明确的说明。
- 活动图可作为未来测试的参考。

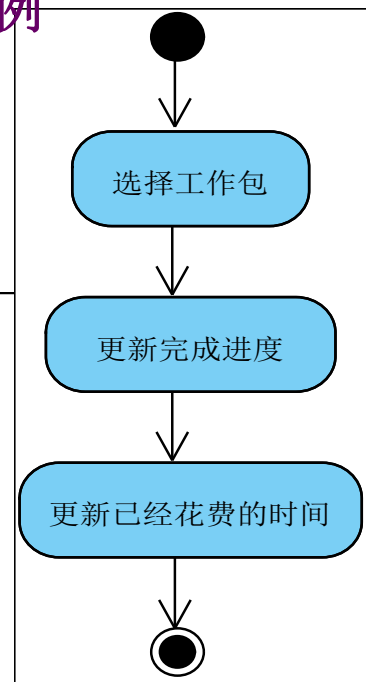
“项目状态分析”用例的活动图



“员工数据维护”用例的活动图



“更新工作状态”用例的活动图



分析过程

用户需求

一家工厂的采购部每天需要一张定货报表，报表按零件编号排序，表中列出所有需要再次定货的零件。

对于每个需要再次定货的零件应该列出下述数据：零件编号、零件名称、定货数量、目前价格、主要供应者和次要供应者。

零件入库或出库通常由仓库管理员完成，通过放在仓库中的CRT终端录入出入库单，完成零件的库存清单管理。

当某种零件的库存数量少于库存量临界值时，库存系统自动记录该零件的信息到订货信息表中。

服务器每天18:00自动扫描订货信息表，并进行分类、累加、排序等整理工作，最后打印出该日的采购单。

	Candidate Class	Extracted Text	Type	Class
1	定货报表	定货报表	Class	
2	仓库管理员	仓库管理员	Actor	
3	采购部	采购部	Actor	
4	需订货零件	再次定货的零件	Class	
5	排序	排序	Use Case	
6	出库	出库	Use Case	
7	录入	录入	Use Case	
8	分类	分类	Use Case	
9	累加	累加	Use Case	
10	整理	整理	Use Case	

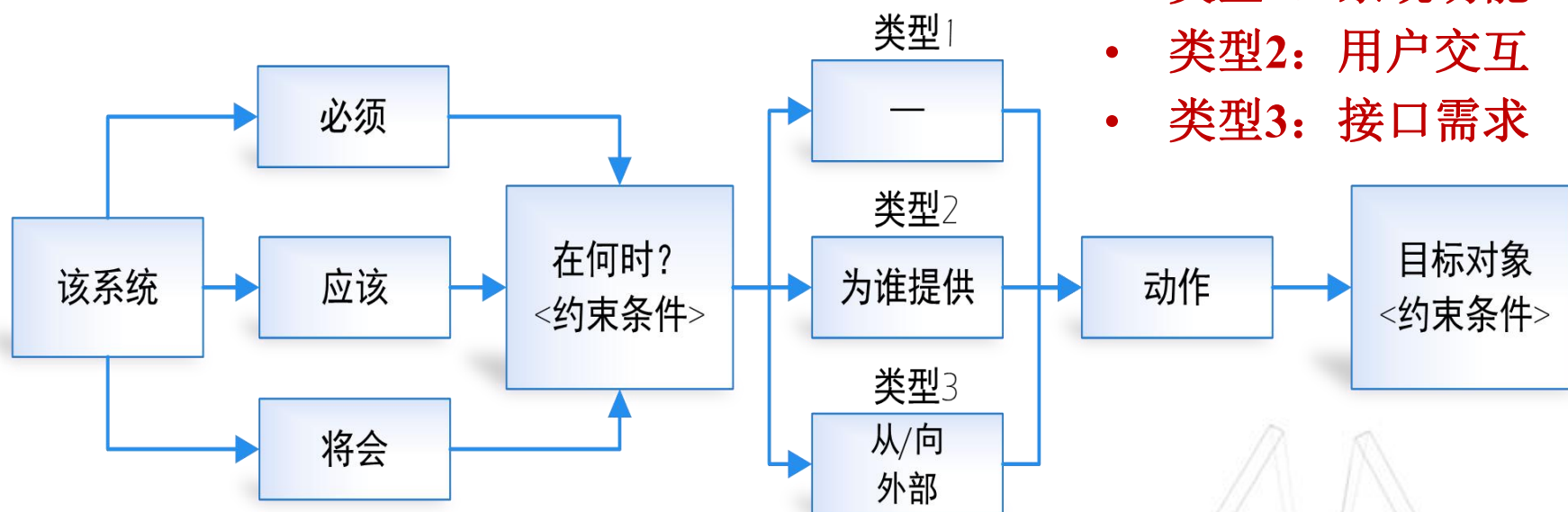
1. 寻找用例
2. 对用例使用模板进行规约并文档化
3. 使用活动图对基本流和备选流进行描述
4. 导出文字表述的功能性需求（系统需求）

需求分析的挑战

- 隐含的假设：领域专家为了简化而遗漏熟悉的内容。
- 笼统的注释：信息被简化的总结。
- 模糊的概括：能否使用“总是”或“从不”等真实的描述业务。
- 迷惑的命名：业务术语以及与其相关的业务动作的命名都需要进行解释和说明。



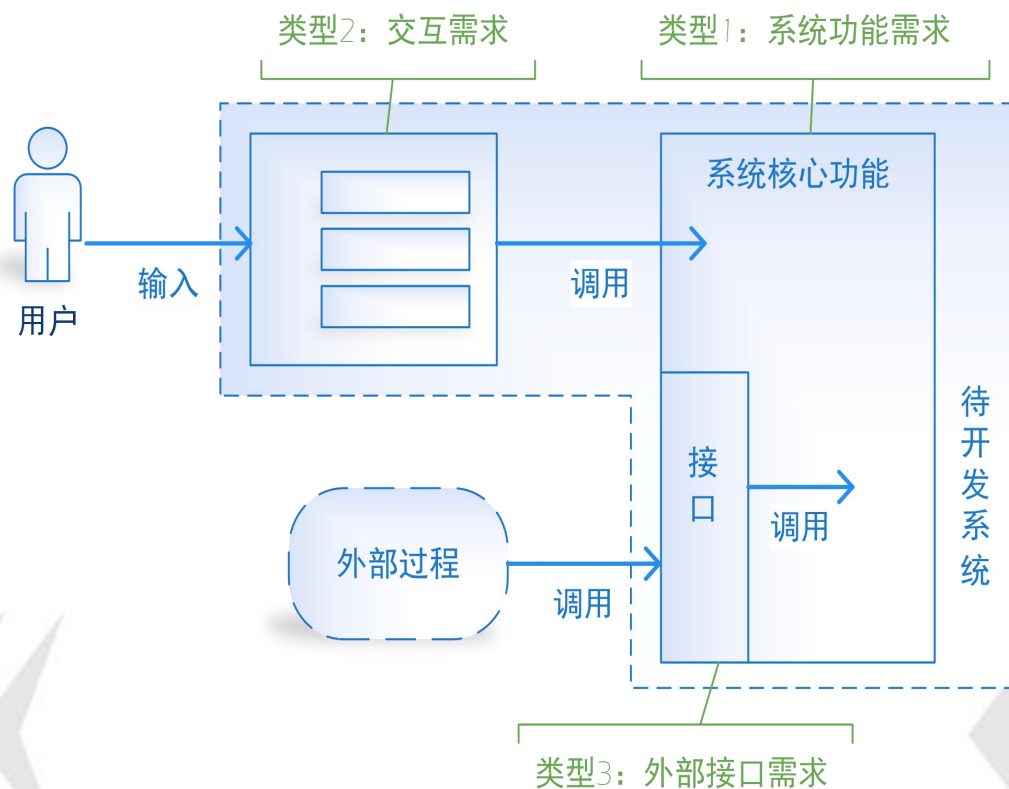
文字需求的模板



- 类型1：系统功能
- 类型2：用户交互
- 类型3：接口需求

- 由活动图派生文本形式的需求描述需要对每一个动作在一个或多个需求陈述中具体化；
- 而且对于每个迁移（箭头）和每个判断都至少要与一个需求对应，或者作为约束条件在某个需求中记录。

三种功能性需求



- 第一类的需求是系统功能的主要组成部分，同时一般也是类型2和类型3需求功能的发起者。
- 第二类一般是提供数据的输入，然后执行对深层功能的调用，并显示出返回的结果，这些深层功能一般为第一类功能。

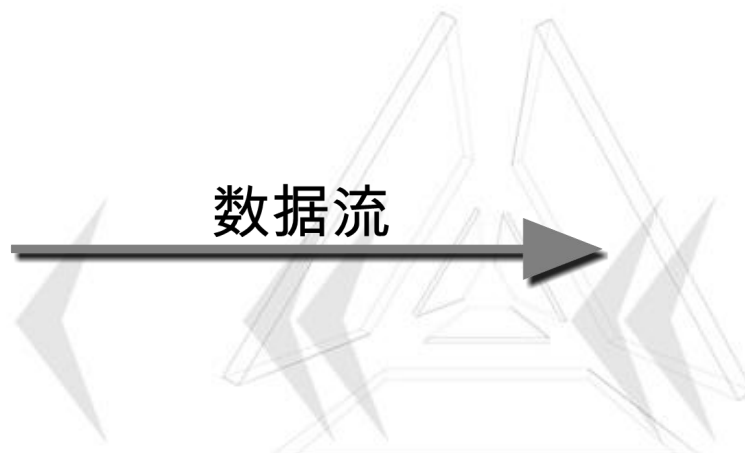
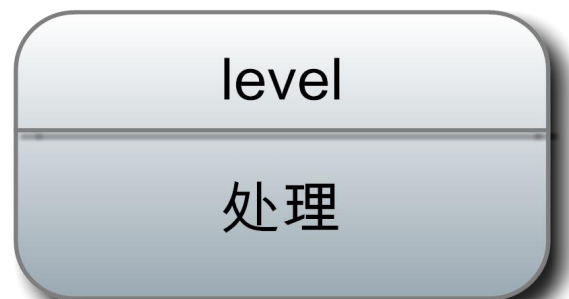
举例

- **R1.1 项目创建**：在项目编辑中系统必须提供给用户新项目的创建以及为其指定具体项目信息的功能。
 - 词汇“项目信息”：自动生成的唯一项目编号、项目名称、项目起止时间、预计工作量。
 - 词汇“项目”：项目可由多个子项目构成，同时子项目也可作为单独的项目。除子项目外，项目还可以包含具体的任务。
 - 词汇“子项目”：项目与子项目在本系统中作为专业术语可理解为同义词。

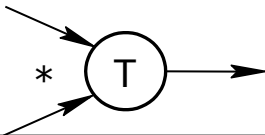
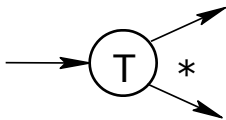
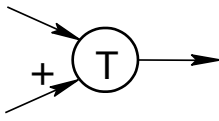
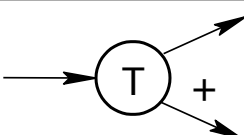
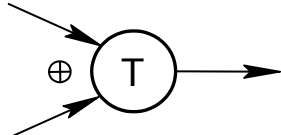
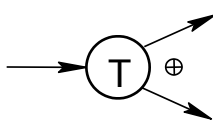
数据流图（传统功能分析）

- 数据流图(DFD) 描绘信息流和数据从输入移动到输出的过程中所经受的变换。
- 数据流图有四种基本符号：
 - 数据的源点或终点；变换数据的处理；数据存储；数据流。
 - 数据流与程序流程图中用箭头表示的控制流有本质不同，千万不要混淆。

数据流图中的基本符号



加工中常见关系的符号表示

符 号	含 义
	由数据 A 和 B 共同变换为数据 C
	由数据 A 变换为数据 B 和数据 C
	由数据 A 或 B ，或者数据 A 和 B 共同变换为数据 C
	由数据 A 变换为数据 B 或 C ，或者同时变换为数据 B 和 C
	由数据 A 或 B 其中之一变换为数据 C
	由数据 A 变换为数据 B 或 C 其中之一

例子：需求

需求陈述

一家工厂的采购部每天需要一张定货报表，报表按零件编号排序，表中列出所有需要再次定货的零件。

对于每个需要再次定货的零件应该列出下述数据：零件编号、零件名称、定货数量、目前价格、主要供应者和次要供应者。

零件入库或出库通常由仓库管理员完成，通过放在仓库中的CRT终端录入出入库单，完成零件的库存清单管理。

当某种零件的库存数量少于库存量临界值时，库存系统自动记录该零件的信息到订货信息表中。

服务器每天18:00自动扫描订货信息表，并进行分类、累加、排序等整理工作，最后打印出该日的采购单。

外部实体

处理

数据流

存储

其他信息

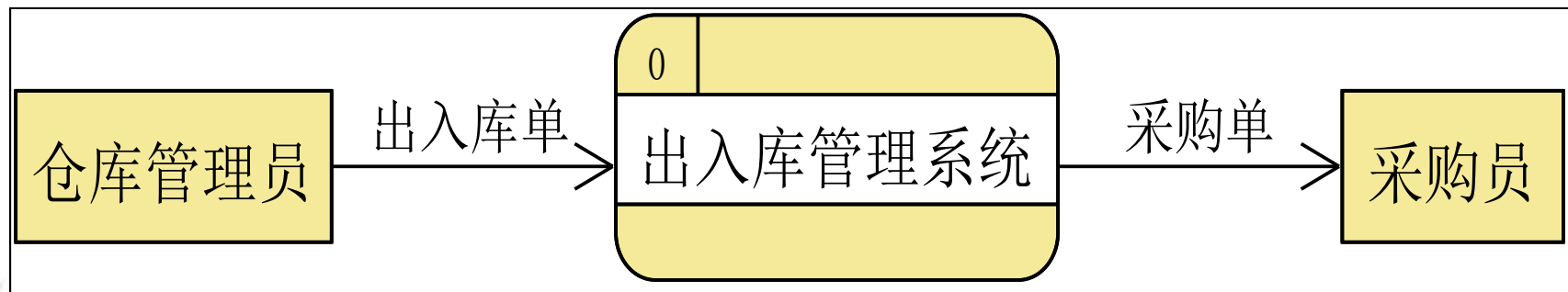


数据流图画法

- 从基本系统模型高的抽象层次开始画数据流图。
- 把基本系统模型细化，描绘系统主要功能。
- 在图中给处理和数据存储加编号，便于引用和追踪。
- 分层细化时保持信息连续性。

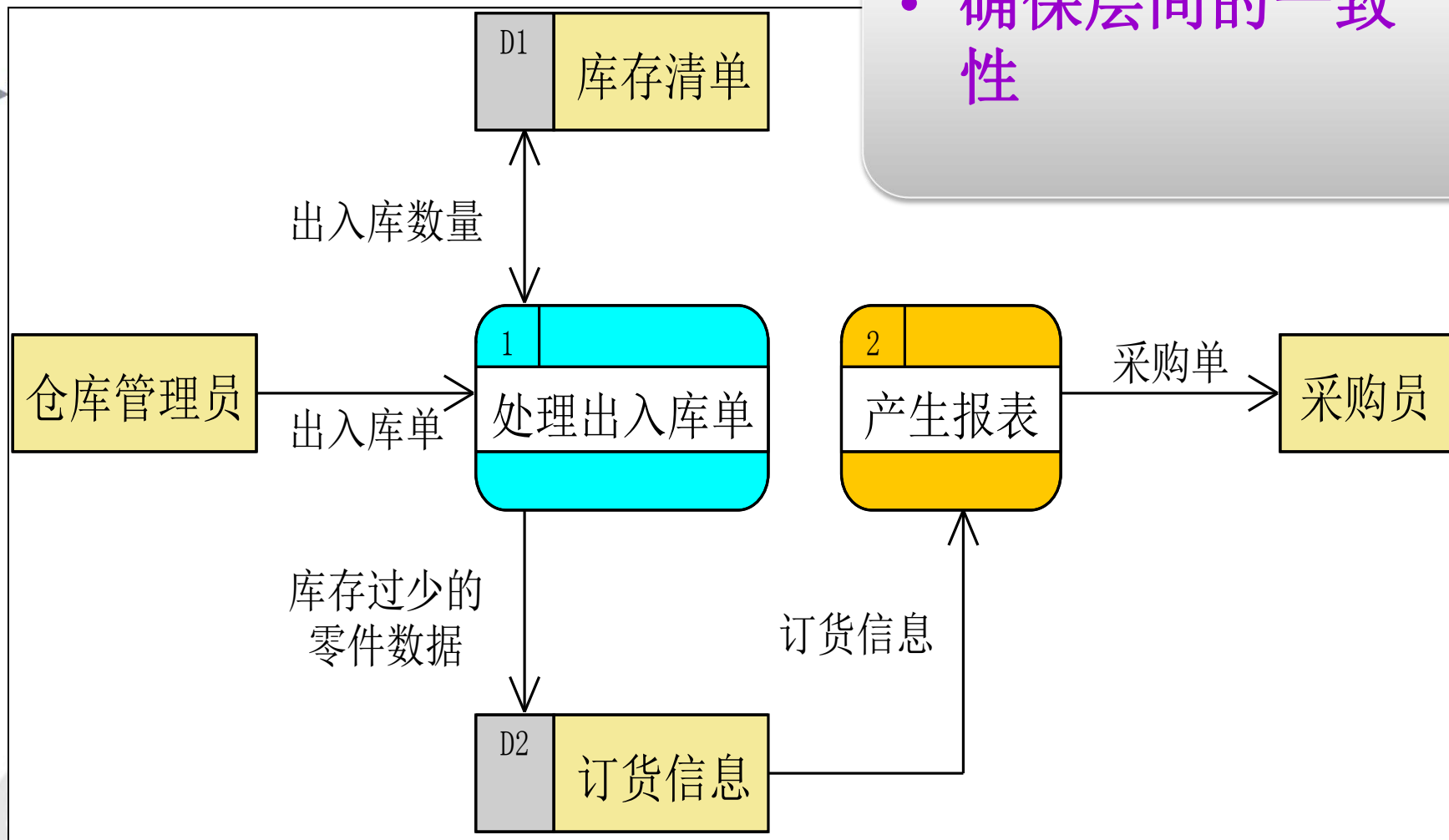
数据流图画法实例

- 信息的流动
- 信息流不能为动词
- 实体到实体的信息流动
- 处理要有输入输出
- 处理名不能为名词

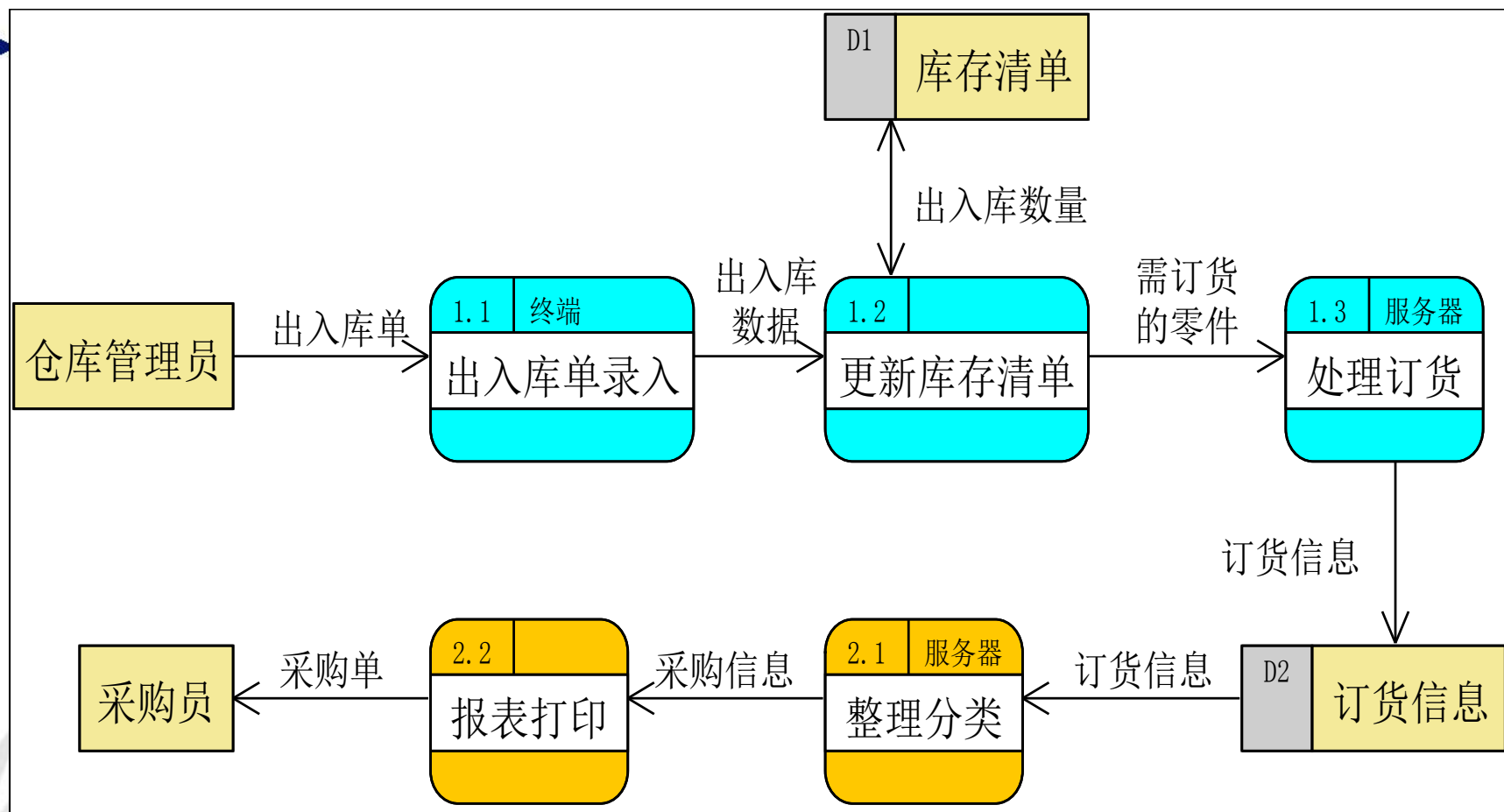


0级数据流图
(突出表明了数据的源点和终点)

- 确保层间的一致性



I级数据流图

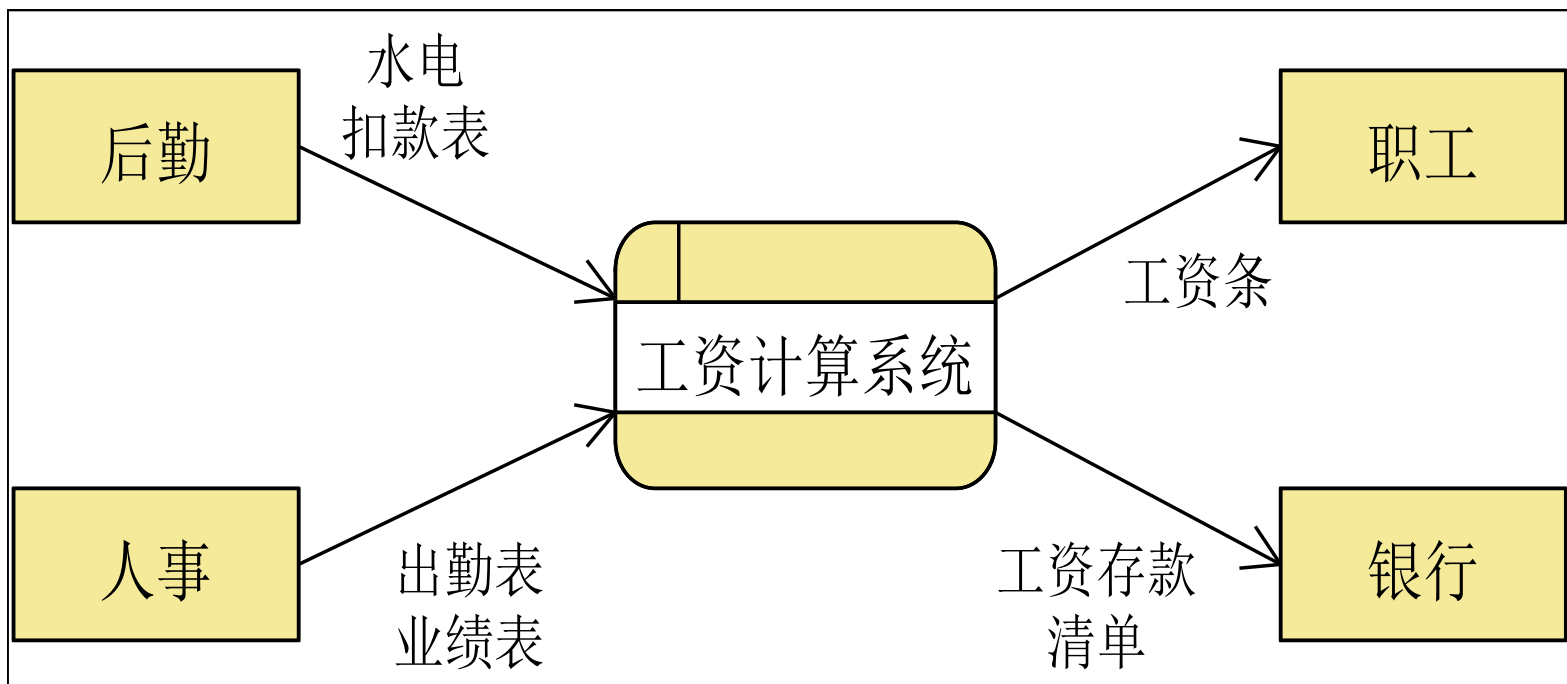


II级数据流图

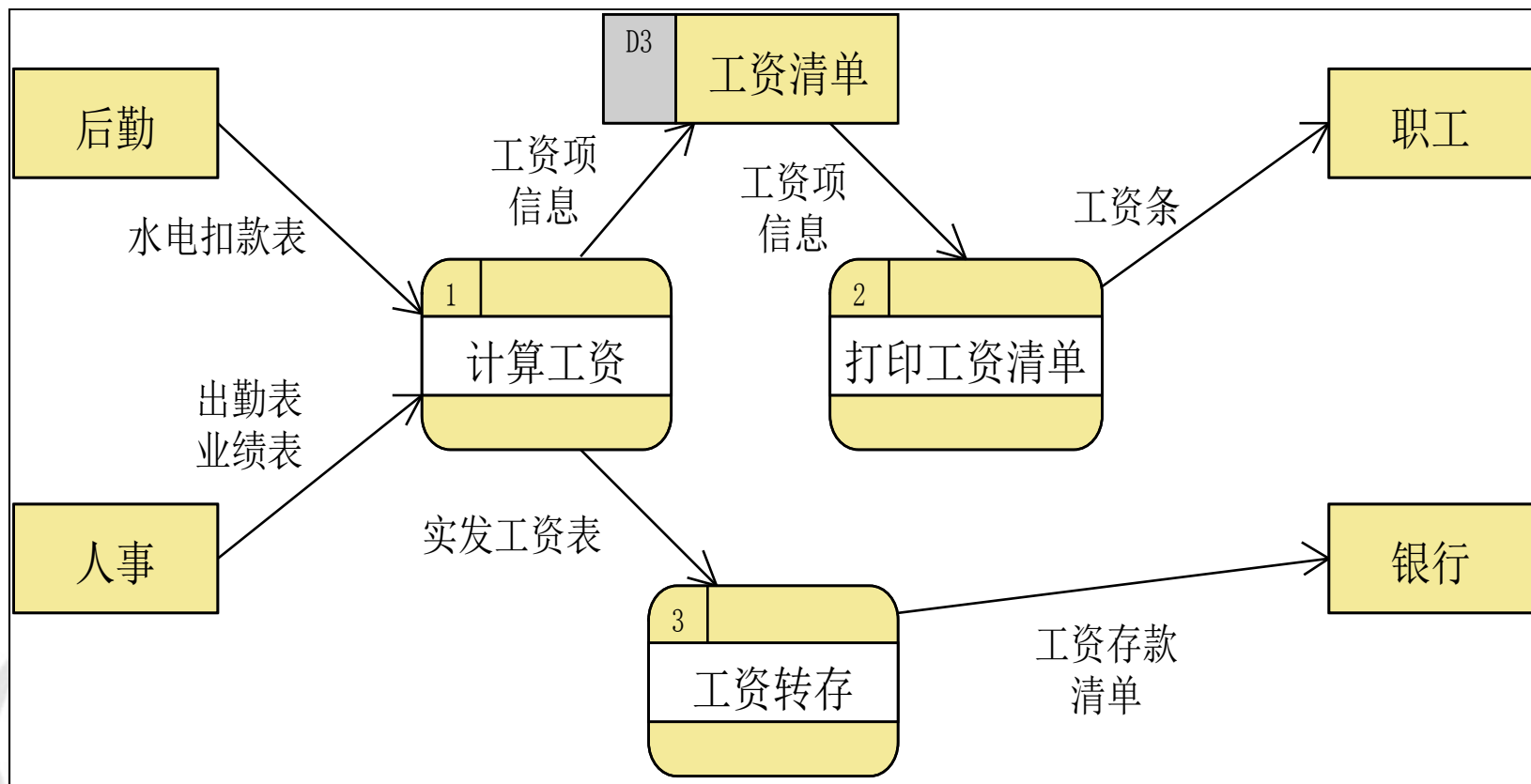
命名

- 应代表整个数据流(数据存储、处理)的内容，而不是仅仅某些成分。
- 不要使用空洞的、缺乏具体含义的名字(如“数据”、“信息”、“输入”之类)。
- 处理名字最好是一个具体的及物动词。

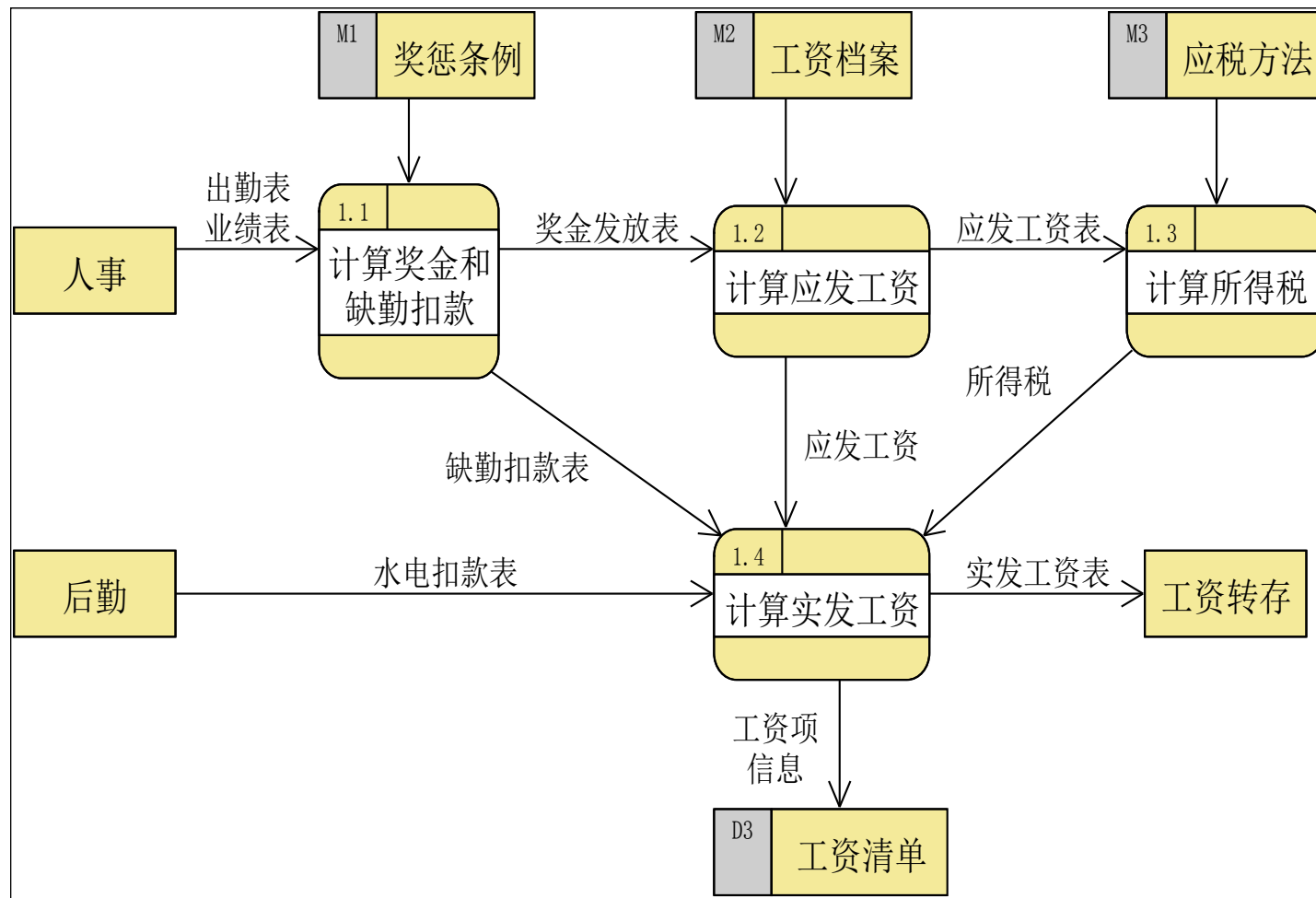
工资计算系统-0级



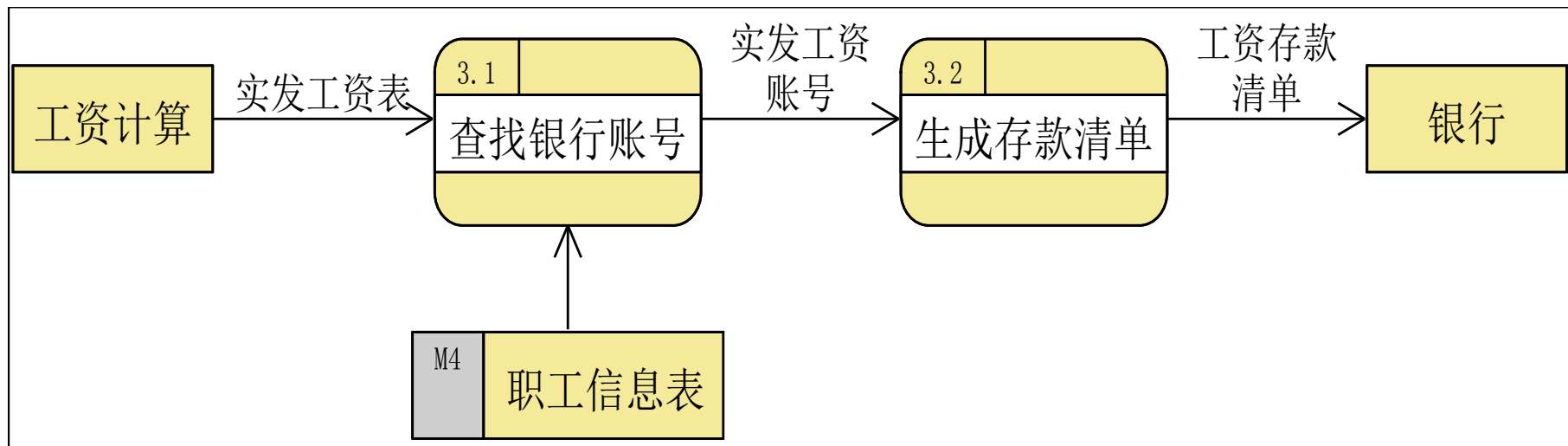
工资计算系统-I级



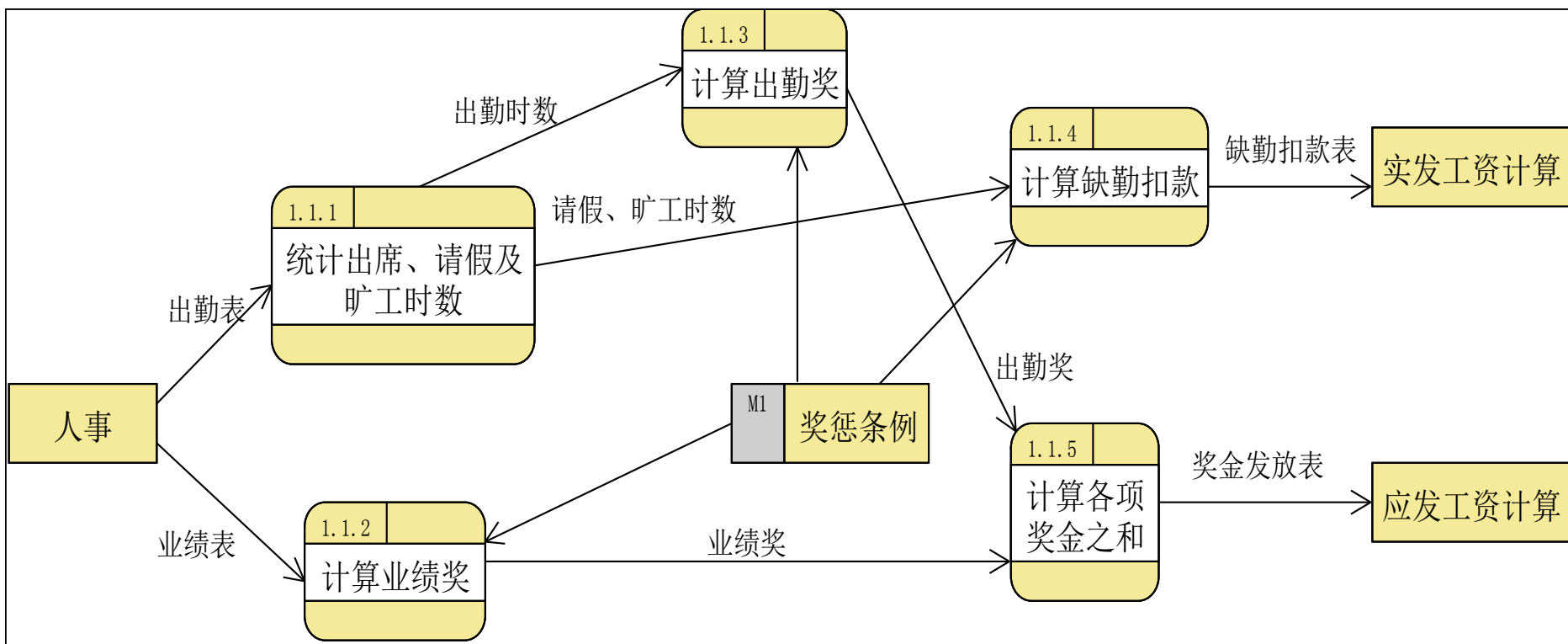
II级-计算工资



II级-工资转存



III级-计算奖金和缺勤扣款



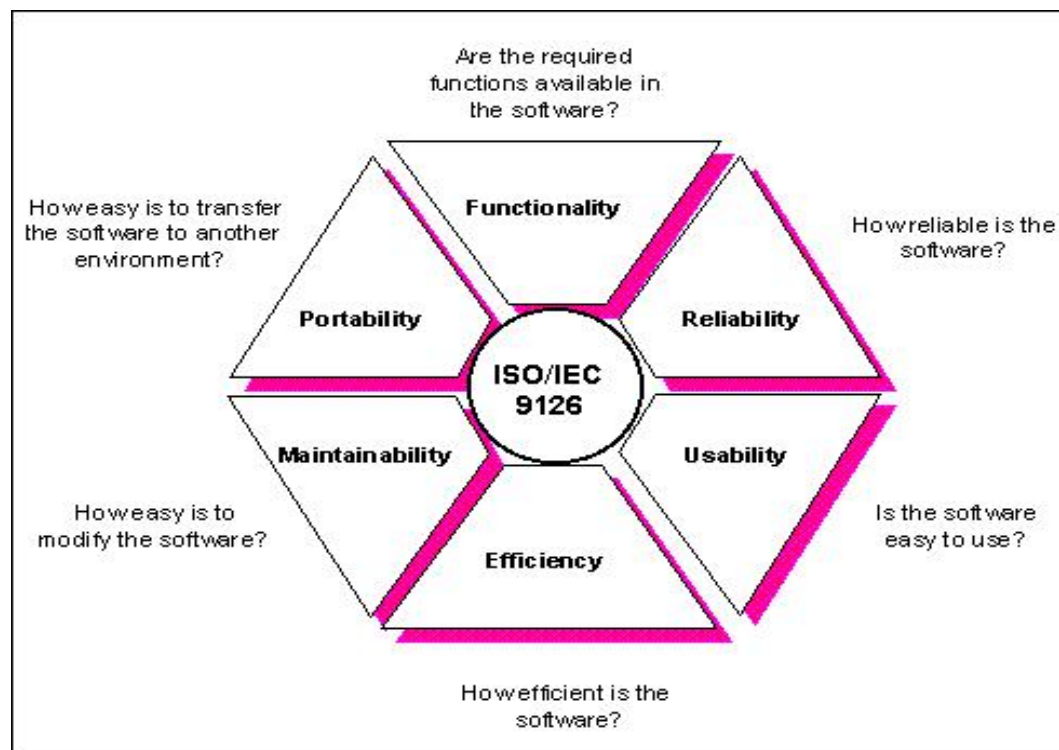
DFD用途

- 交流信息的工具
 - 一张数据流图处理少于9个
 - 分层
- 分析和设计的工具
 - 在数据流图上划分自动化边界
 - 每个边界意味着不同的物理系统。

- 系统流程图：物理构成
- 数据流图：逻辑功能
- 面向数据流图的设计方法

非功能性需求

- 一些利益相关者没有纯功能上的需求，但要满足他们对最终软件产品某些方面的要求，比如开发规范等。
 - 质量需求
 - 技术性需求
 - 硬件要求
 - 软件平台
 - 运行环境
 - 其它交付物
 - 合同需求
 - 商业条款
 - 规格说明书
 - 技术协议



需求说明书

0. 文档说明性内容

- 版本号, 创建者, 修改记录, 批准者

1. 目标群体和系统目标

- 利益相关者分析、项目公开和隐藏的目的

2. 功能性需求

- 所有功能性需求及其用户故事, 从建立在活动图上的用例到文字性的需求描述, 文档化

3. 非功能性需求

- 特别是质量需求及技术需求

4. 交付物

- 具体时间和形式交付的产品

5. 验收标准

- 对需求进行检验的方法以及结果处理方式

6. 附件

- 所有相关文档的列表, 包括需求分析阶段的词汇表

需求验证的方法

- 验证需求的一致性
 - 自然语言书写：人工技术审查
 - 形式化描述：软件工具
- 验证需求的现实性
 - 开发经验、仿真或性能模拟技术
- 验证需求的完整性和有效性
 - 用户验证
 - 原型

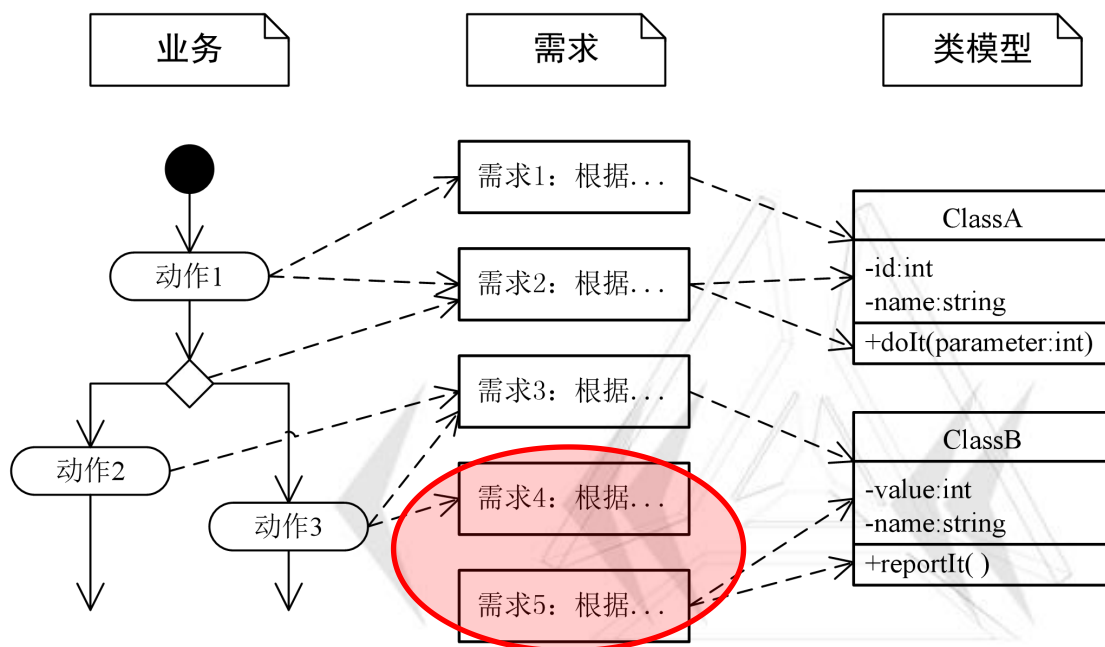
大规模的项目比较困难，多采用风险模型挑选重要部分进行详细审查。

需求跟踪

- 给出针对每个源需求及对应目标实现，将它们通过某种方式联系起来。

- 从用例图到活动图的转换
- 从活动图或者动作（**action**）到文本需求的转换
- 从文本需求到类或者实例变量、方法和方法参数的转换

- 文档记录的方法，大规模系统难以满足要求。
- CASE工具。**



跟踪的作用

- 需求变更影响范围如何
- 从类逆向追溯需求，可以获知实现的改动会影响到哪些原始需求
- 除此之外，跟踪信息还可以附加对应连接创建的时间、属于增量开发的哪个周期等
- 测试用例和需求之间的跟踪连接，可以方便地确定出哪些需求已经进行了测试，哪些还存在遗漏，对于掌握和提高测试的完备性具有重要的指导意义
- 在CMM三级中要求软件组织必须具备需求跟踪的能力

需求跟踪矩阵

- 实践中常使用需求跟踪矩阵（RTM）对变更进行管理，包括需求变更、设计变更、代码变更、测试用例变更等
- 需求跟踪矩阵是变更波及范围影响分析的有效工具
- 可将RTM划分为两类：**纵向跟踪矩阵和横向跟踪矩阵**。
- 纵向跟踪矩阵，包括如下的3种跟踪内容：
 - 需求之间的派生关系，如客户需求到系统需求；
 - 实现与验证的关系，如需求到设计、需求到测试用例等；
 - 需求的责任分配关系，需求由谁负责。
- 横向跟踪矩阵包括需求之间的接口关系等。
- 需求跟踪矩阵只要能够保证需求链的一致性和状态的跟踪就可以，有不同的实现方法。

作业

- 习题1~5

