软件需求最佳实践 SERU 过程框架原理与应用

作者： 徐锋 [作译者介绍]
出版社：电子工业出版社
ISBN：9787121073953
上架时间：**2008-10-27**
出版日期：**2008** 年 **10** 月
开本：**16** 开
页码：**396**
版次：**1-1**
所属分类： 计算机 > 软件工程及软件方法学 >

作者：徐锋

徐锋：高级程序员、系统分析员(高级工程师)，中国系统分析员顾问团(CSAI)华东区首席顾问，《程序员》杂志社专栏作者。具有丰富的 IT 市场规划、技术管理、产品研发、需求分析、产品建模、体系架构设计、软件开发等工作经验。现在致力于系统分析与设计、需求工程、软件过程改进等领域的研究。已在《程序员》、《中国计算机世界》等报刊杂志上发表各类文章近 50 篇，其中实战 OO、大话 design 等专栏都深受读者喜爱与好评，参与 / 独立编著出版的图书近 20 本。. 作者亲历了各个级别的软考，有丰富的考试、培训与教程编写经.. << 查看详细

本书首先从软件需求实践中出现的主要问题和困难入手，指出了改进的主要方向；然后逐一说明了需求定义、需求捕获、需求分析与建模、编写规约、需求验证等需求开发活动的任务、要点和具体手段；并提出了一个可操作性强、易于上手的 seru 过程框架，能够帮助读者清晰地了解整个过程，理解各阶段的关键产物和产物之间的关系。

本书还对包括需求基线、变更管理、需求跟踪在内的需求管理活动的操作要点进行了阐述，给出了具有很强实践性的具体建议。综观全书，语言浅显、文字生动，蕴含了许多人文、心理、交流方面的知识，即使非技术背景的读者也能够轻松读懂大部分内容，从中受益。

本书可作为计算机软件专业本科生、研究生和软件工程硕士的软件需求分析教材，也可以作为软件工程、软件开发管理培训的教材，更是一线项目经理、需求分析人员、资深开发人员、信息系统运行管理人员、研发企业管理人员的必备参考书。

第 1 部分 原理、模型与误区

第 1 章 需求实践现状分析

1.1 软件项目失败的根源

1.1 软件项目失败的根源

1.1.1 chaos report 1994

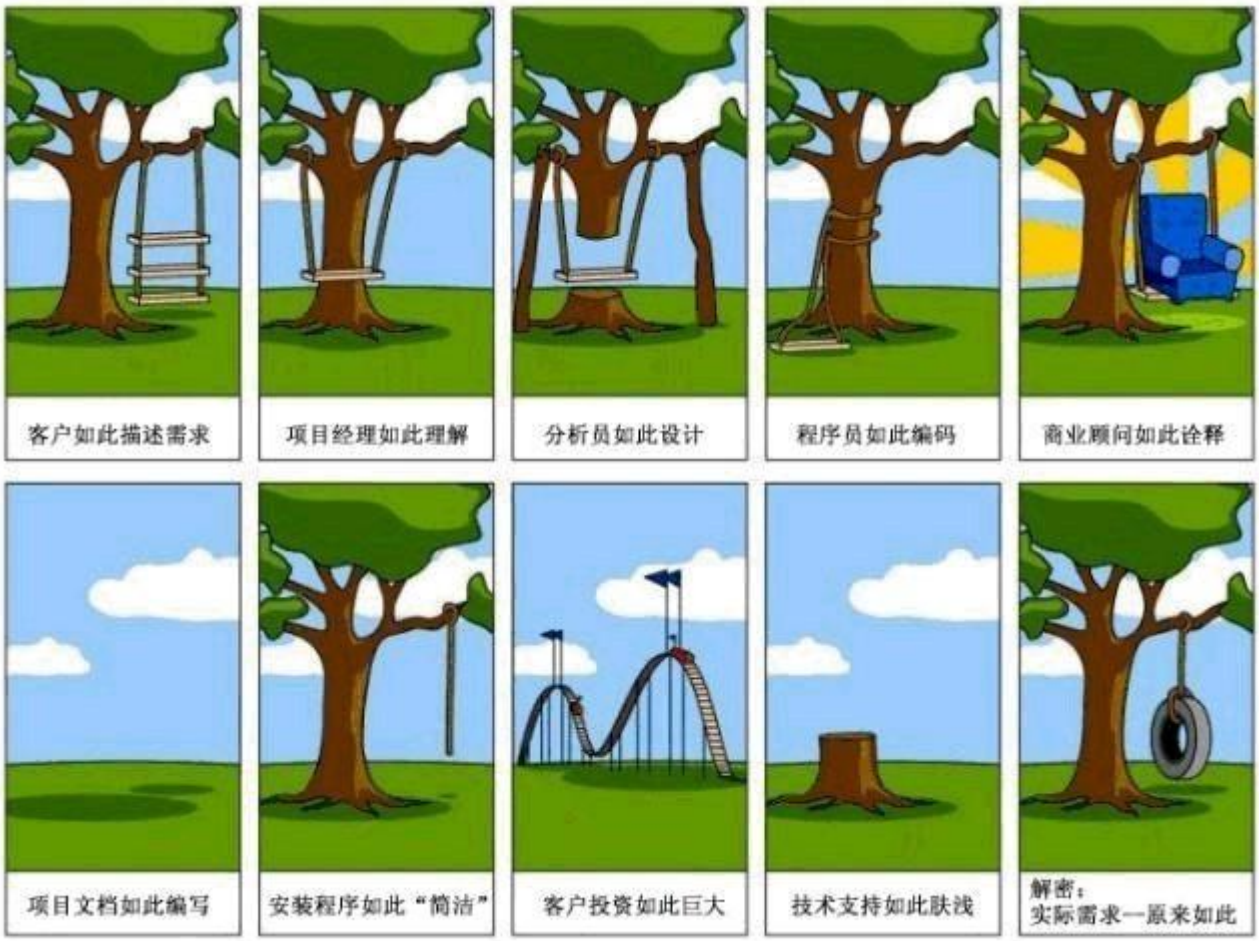
1.不完整的需求
2、缺乏用户参与
3、资源不足
4、不切实际的用户期望
5、缺乏执行层的支持
6、需求的变更频繁
7、规划不足
8、提供了不再需要的
9 、缺乏 IT 管理
10、技术能力缺乏
11、其他

1.1.2 chaos report 后续版本

1.1.3 需求相关败因简要分析

败因	解决方案:
1 不完整的需求	采用业务导向让用户参与到完整性评价当中，在实际过程中利用树形层次结构将宏观信息与微观信息进行有效的剥离。 分析: 利用树形层次结构面向不同的层面：决策层，事务管理层，操作层，让合适的人验证相关的部分，这样可以有效的避免事不关己，高高挂起。让用户逃离无趣区和有效的启发用户的积极性，减少被专业人士排除在系统的完整性评价之外
2 缺乏用户参与	
3 不切实际的用户期望	由于软件的无形和成本的不透明，需要说明软件中不能完成的需求的原因
4 需求变更频繁	对变更进行分类，统计，有针对性的改进需求过程，提高设计的弹性
5 提供不在需要的	找到用户经常使用到的功能，也即用户的需求，放弃用户很少使用的功能模块的开发 具体方案：在每个功能模块入口处，放置一个计数器

1.1.4 一幅漫画带来的思考



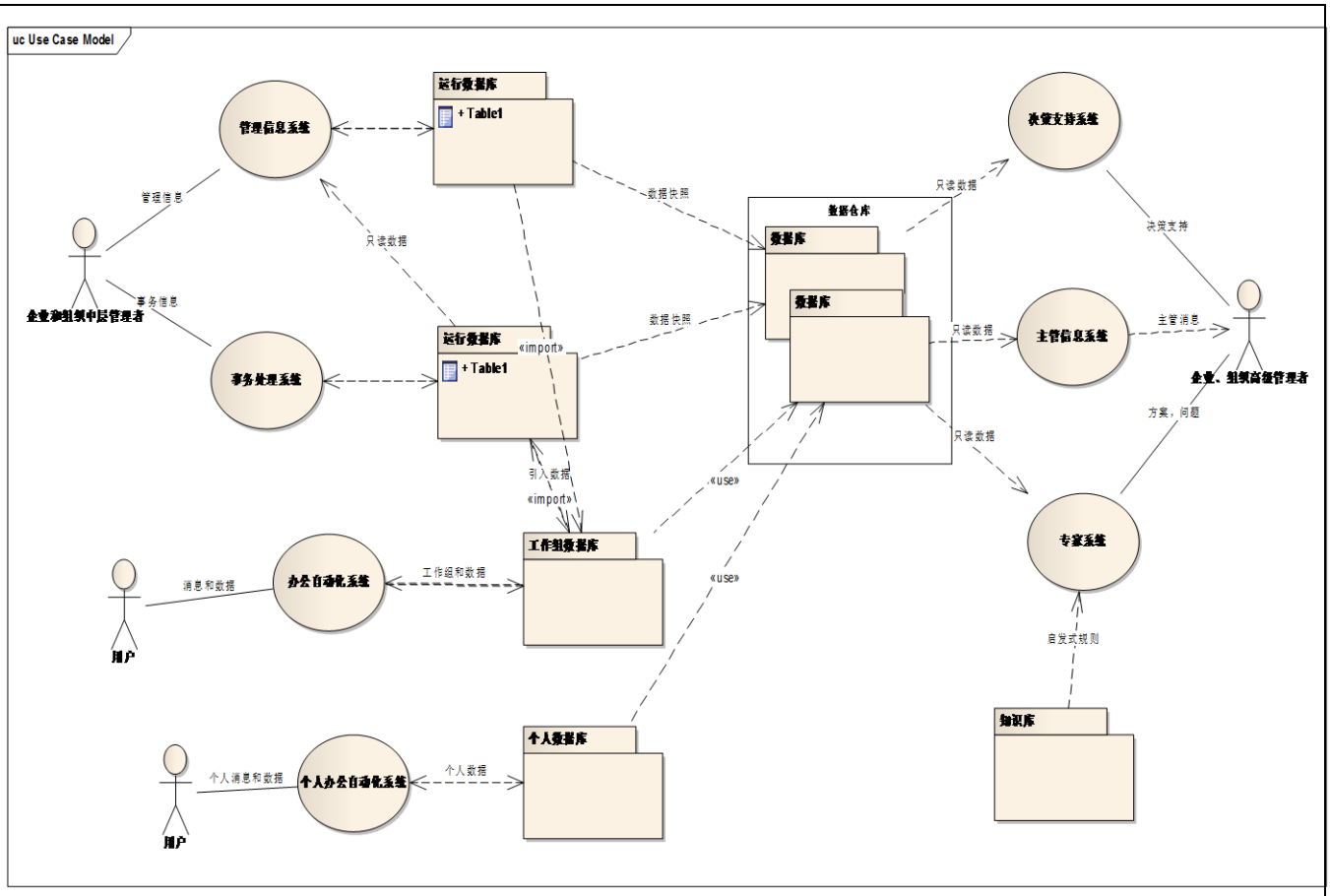
失败的原因	分析原因与解决方案
1 沟通失真	分析： 每个角色（如项目经理，商业顾问等）更加自己的特点和需求对信息（漫画）进行了不同程度的加工，从而导致信息内容有很大的变化
	解决方案： 1、 利用文档：防止信息在传递的过程中走样 2、Reviews （评审）：在此审阅，需求人员在此用语言复述软件需求。
2、客户需求放大	分析： 1、客户希望支付的成本尽可能少，获得的效益尽可能多。 2、解决方案的选择权较给了不熟悉技术的客户。
	解决方案 1： 1、需要提升软件估算实践的有效性 2、提高产业成熟度 解决方案 1： 需求捕获的过程中多问为什么
3 项目经理的需求控制	分析： 需求捕获的过程中，导致需求产生了偏差，部分从技术的角度对需求

	进行了控制，造成无法对需求的有效理解
	解决方案： **减少技术在需求提取过程中的不利影响**
4 分析人员的技术加工	**分析：** **分析人员对技术能力的追求高于业务能力的追求**
	解决方案： **提高自己的业务分析能力**
5 编码人员的断章取义	**解决方案：业务场景是需求之魂**

1.2 透过表象，分析本质	
1.2.1 需求变更频繁	
1.2.2 上线阻力大	
原因	解决方案
1 利益冲突	需求分析
2 工作量大	提高易用性、工作量价值化、将数据迁移，准备工作独立出来
1.2.3 运行效果差 解决方案：从问题与机会入手，提高管理人员的推动力 从障碍和困难出发，解决操作人员的积极性	
1.2.4 完全崩溃 分析：大多是由于忽略了非功能性需求（如：系统容量不足以支撑业务需求 解决方案：明确非功能性需求的特点,然后进行改进	
1.3 方法论与需求工作	
1.3.1 计算模式	

如：C/S 与 B/S	从用户的角度选取
1.3.2 软件工程方法论	
方法论分两种：重方法和敏捷方法论	
1.3.3 开发思想	
1 成熟度	判断标准之一：见报率高，成熟度低
2 溯源	了解本质
3 了解局限性	了解局限性才不至于滥用，误用
1.4 小结	

第 2 章 不同软件项目的需求视图
2.1 信息系统的需求视图



2.1.1 信息系统的本质与分类

信息系统的几个要素

1 支持企业日常运作	
2 支持解决问题	解决企业运作中存在问题的使命
3 支撑决策	加工处理数据
信息系统的本质	
1 数据信息化	通过对数据的有效处理，从而得出对人们更有价值的信息

2 信息系统的类别

联机事务处理系统(OLTP)——数据的生产者
管理信息系统(MIS)—数据的消费者（企业、组织中层管理者用于处理事务）
主管信息系统(EIS)—数据的高级消费者（企业中的高层管理者用于决策）
决策支持系统（DSS）——数据的高级消费者（中层管理者）
专家系统——对个人知识的沉淀，同时也是数据的消费者
办公自动化系统（OA）——对沟通和协作的支持

2.1.2 联机事务处理系统——流程电子化

电子化流程的好处	1、有利于流程的固化 2、对业务产生一定的约束

2.1.3 管理信息系统——数据信息化

1 报表需求何时开始分析

（1）OLTP 是数据的生产者，MIS 是数据的消费者

报表的数据从 OLTP 中来，传统需求实战中，我们却经常先分析业务功能性需求，在分析报表类需求，此时我们根本不了解消费者需求是就确定了生产者的需求，显然是一种草率的态度。

解决方案：

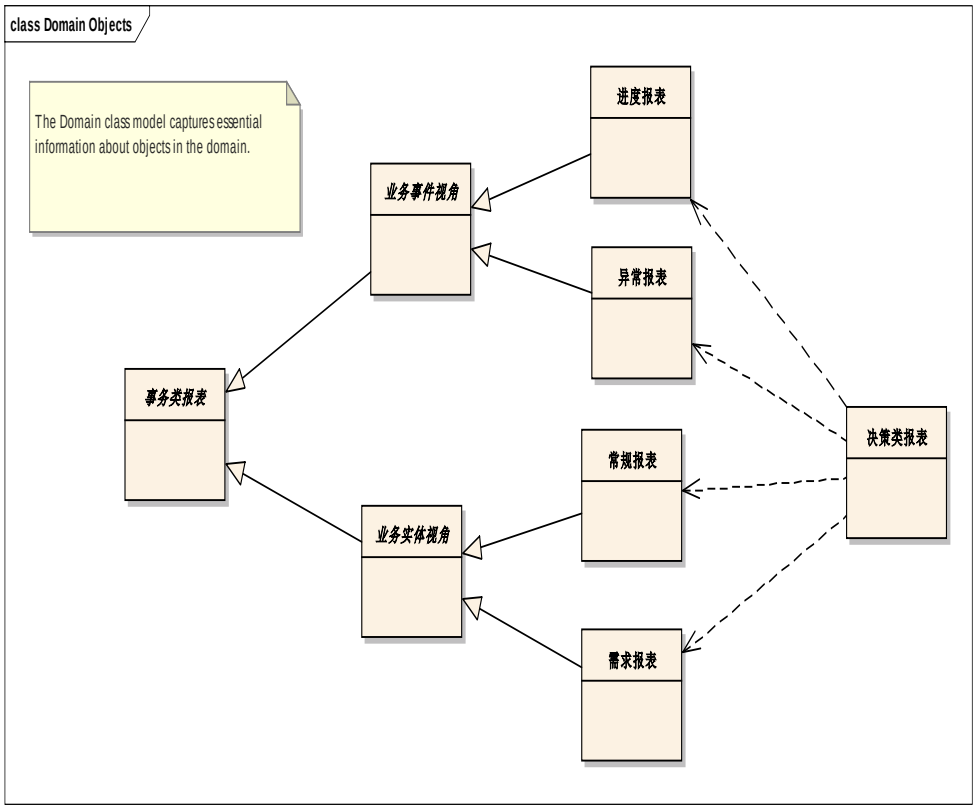
从管理场景入手，从理解报表的目的着手就能更好地完成与用户的沟通，而且也更加的高效。

报表需求的要点：	Why	目的	从管理场景出发，借助对管理控制点的理解来理解报表的目的
		使用部门/职位	了解使用者
		相关场景	如用户数量
	What	关联实体	以类图或 E-R 图说明数据的来源
		关键指标及计算规则	细化推导出关联字段，以及派生属性的基本方法
	How	展现形式	以虚拟窗口等形式
		输入输出需要	说明是否打印，格式等

2 解析报表的分类

1 事务类

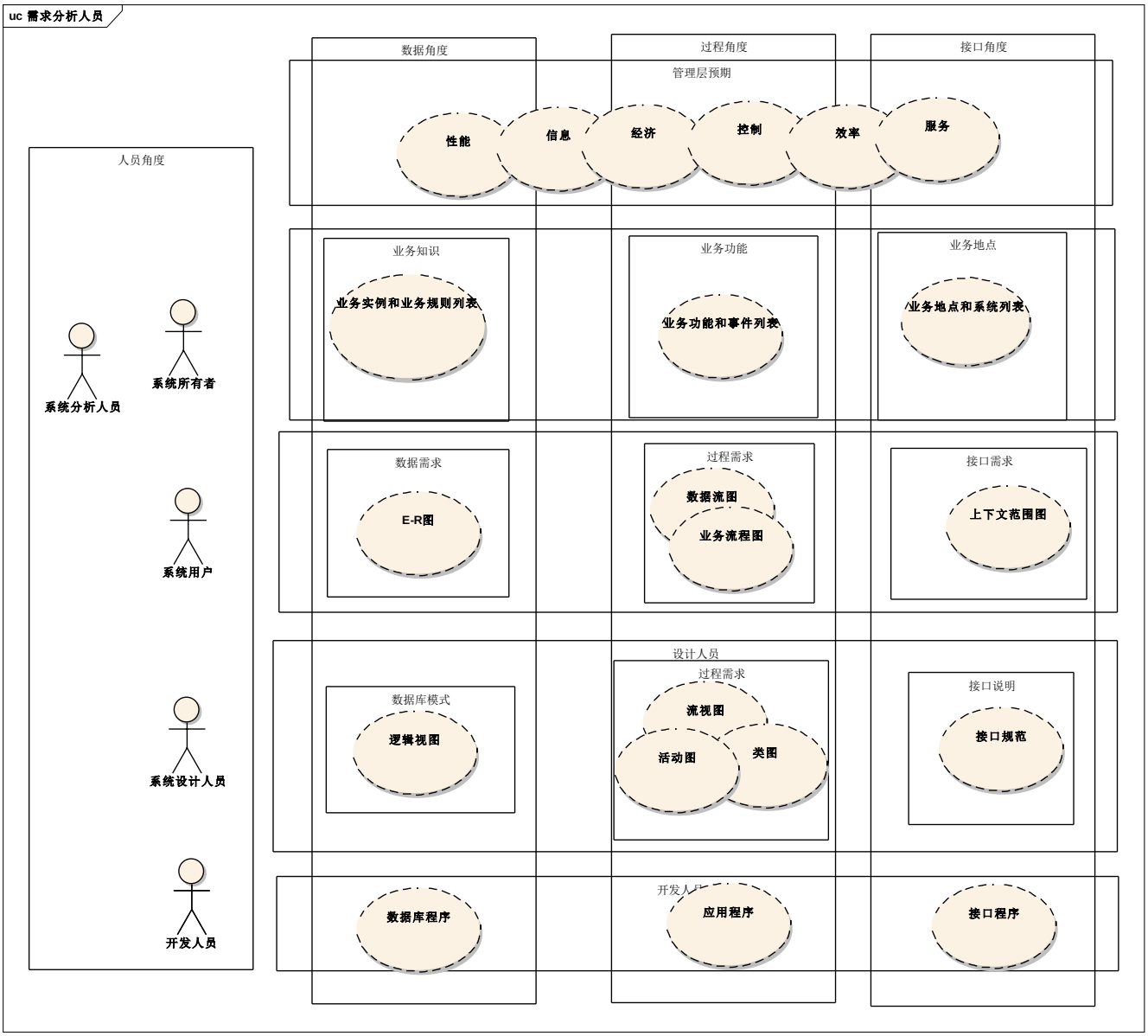
2 决策类



2.1.4 其他信息系统

1 决策支持系统	1、结构化问题： 利用历史积累经验，如安全库存量，现金流预警。可以建立模型利用计算机直接计算出来。 2、非结构化问题； 如广告投放，产品定位等问题都没有现成的模式可以依赖
2 专家系统	个人知识转化为企业知识
3 办公自动化	协同信息化

2.1.5 信息系统的多维视图



信息系统多维视图

人员角度		数据角度	过程角度	接口角度
需求分析人员	系统所有者	业务实例和规则列表 业务知识	业务功能和事件列表 业务功能	业务地点和系统列表 业务地点
	系统用户	E-R 图 数据库需求	数据流图和业务流程图 过程需求	上下文范围图 接口需求
	设计人员	逻辑视图 数据库模式	流程图、类图、活动图 过程需求	接口规范 接口说明
	开发人员	数据库程序	应用程序	接口程序

2.2 嵌入式系统的需求视图

2.2.1 面向直接用户的嵌入式系统

2.2.2 面向特定设备的嵌入式系统

2.3 软件产品的需求视图

1 信息类

（1）、目标市场分析

1、目标客户分析

2、竞争对手分析

3、商业模式分析

（2）、产品体系设计

信息系统类软件产品的需求重点在于针对不同目标客户群体的不同商业模式分离变化点，经常需要减出通用性，在通过插接解决扩展性

工具软件类	对现实世界中某种工具的仿真，例如计算器，便签

2.4 小结

第 3 章 软件需求与需求工程

3.1 什么是软件需求

定义	业务知识+问题列表+其他因素

3.1.1 需求的三个层次

1 业务需求

定义：软件系统的建设目标。

需求定义的产物

也就是反映企业/组织对软件系统的高层次目标要求：

1、问题：解决企业运作过程中遇到的问题，如物资供应脱节、用户投诉量大

<table>
<tr><td colspan="2">2、机会：抓住外部环境所带来的机会，以便为企业带来新的发展，如电子商务，网上银行
业务需求应该在项目立项的时候整理。</td></tr>
<tr><td colspan="2">2 用户需求
定义：用户使用软件需要完成什么任务，怎么完成的需求。
需求捕获的产物
通常在业务需求定义的基础上进行用户的访谈,调查,对用户使用的场景进行整理，从而建立用户角度的需求。
1、零散；用户提出不同角度，层面，粒度的需求
2、存在矛盾：用户在企业中的不同层面导致对系统的需求不同</td></tr>
<tr><td>3 软件需求</td><td>需求分析与建模的产物</td></tr>
<tr><td colspan="2">3.1.2 需求的三种类型</td></tr>
<tr><td>1 功能需求</td><td>以系统→子系统→模块→子模块方式组织</td></tr>
<tr><td>2 非功能需求</td><td>常见的问题：是什么
保证信息的有效传递和注意局限性
1、信息传递的无效性，如高可靠性，扩展性
2、忽略了非功能需求的局部性，如查询时间<10s</td></tr>
<tr><td colspan="2">3 设计约束</td></tr>
<tr><td>1 非技术性因素决定的技术选型</td><td>如：采用 VC++平台</td></tr>
<tr><td>2 预期的软硬件环境和使用环境</td><td>实现技术受环境的影响如内存大小，海上使用</td></tr>
</table>

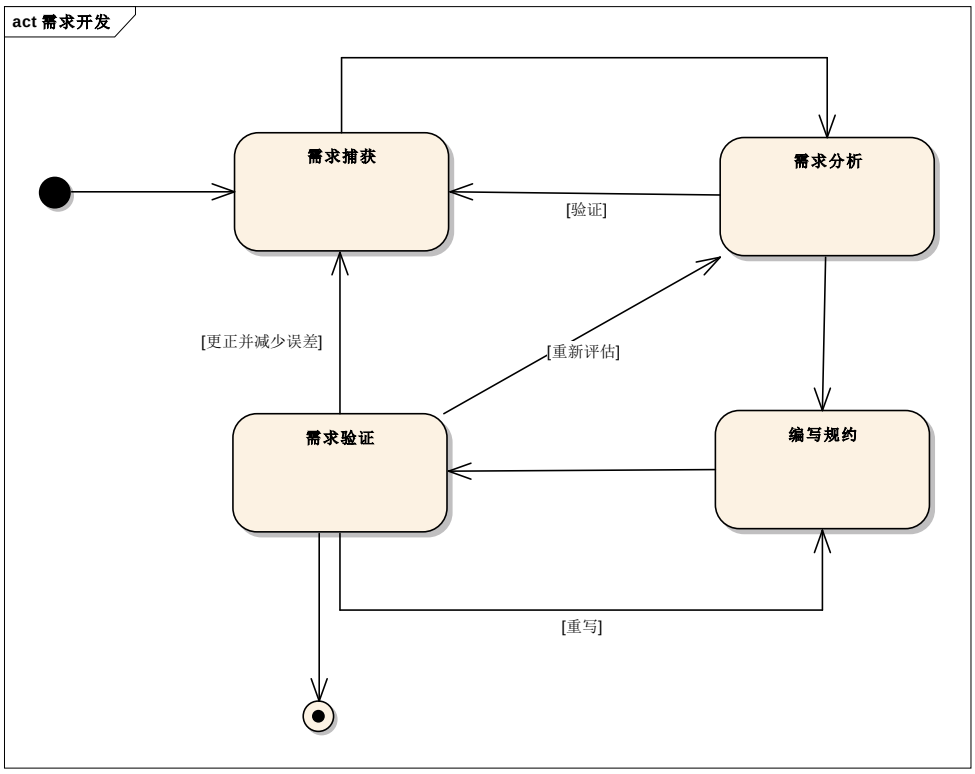
<table>
<tr><td colspan="2">3.1.3 优秀需求的标准</td></tr>
<tr><td>1 完整性</td><td>定义：需求没有遗漏，也就是说在需求变更中新需求都是因为外部环境的变化而产生的且所占量小
（1）用户才是验证需求完整性的合适人选
为了保证需求的完整性，就必须从业务角度来组织各种需求项，让用户验证需求规格说明书中罗列的主题域、业务事件、业务活动、业务步骤、困难与障碍点是否完整，更具操作性。
步骤：验证主题域-》中层-》操作层</td></tr>
<tr><td>2 不失真</td><td>1、正确性
2、无歧义性</td></tr>
<tr><td colspan="2">3 有优先级</td></tr>
<tr><td>1 优先级有不同角度</td><td></td></tr>
<tr><td colspan="2">2 必要性是优先级评价的盲区
必要性只是对优先级的补充
分析:优先级常从充分性来考虑的,这样会导致忽略了需求的必要性。利用满意(反映需求的充分性）\不满意（反映需求的必要性）模型来防止需求的蔓延。</td></tr>
<tr><td colspan="2">4 有技术早期介入
1、可行性；就是指需要让开发团队早期介入，对需求中描述的解决方案进行评价。重点在需求项及复杂的解决方案
2、可验证性；说明需求规格说明书应该能够指导测试活动，也提供了验证所需的信息。</td></tr>
</table>

3.2 需求工程解析

3.2.1 需求工程的范畴	1、需求开发：收集、分析、整理、编写、验证 2、需求管理：对需求的实现变化进行追踪的全过程，重点在于确保开发的软件满足这些需求

需求开发的三次循环

循环	工作任务	对应的 RUP 阶段
初始循环	明确项目目标与范围，完成子系统划分及相关内容和接口	初始阶段
脉络循环	通过对每个 业务事件进行流程分析、业务实体分析，并表示出所有用例	细化阶段的第一迭代
细节循环	对每个用例的细节进行分析，包括事件流，用户界面原型	细化阶段的第二次迭代构建阶段

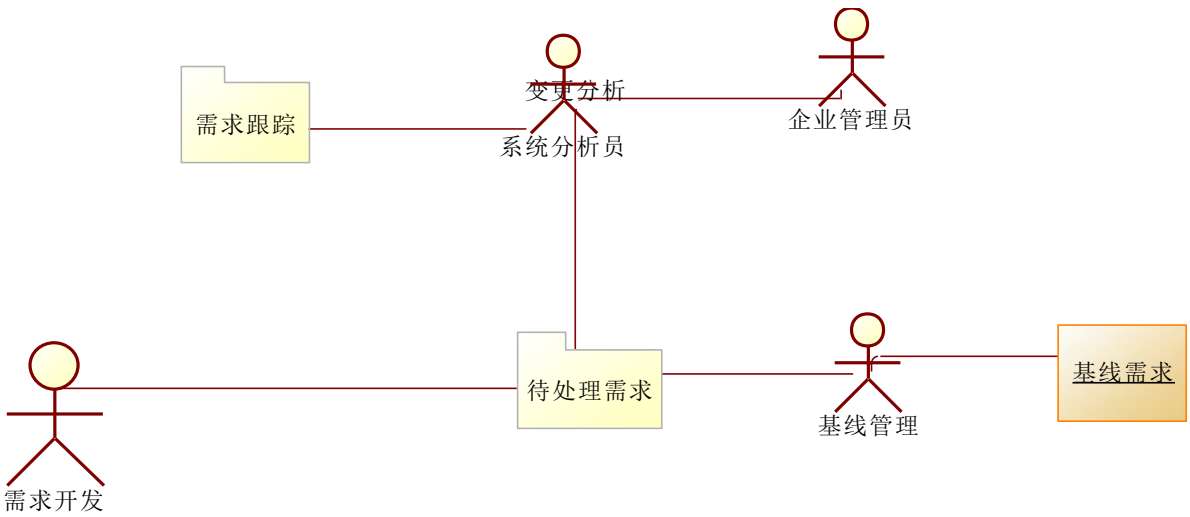
<table>
<tr><td></td><td> 
</td></tr>
<tr><td colspan="2">分析说明：</td></tr>
<tr><td>1 需求获取</td><td>需求的捕获，它们都是主动词，而现实的需求实践中最大的问题是比较被动的。主要体现在：
1、捕获范围不足（对业务知识的不足）
2、缺乏计划性（预先对问题、时间、采访对象没有计划）
3、缺乏科学性（如宏观与微观问题混在一起）
4、捕获对象不明确（很少主动寻找合适的被访者）
5、捕获手段不足（对场景不同时需求的捕获缺乏）</td></tr>
<tr><td>2 需求分析</td><td>1 需求分析是什么
定义：
1、需求分析是业务分析
2、需求分析是一种分解活动
3、需求分析是一种提炼与整合活动</td></tr>
</table>

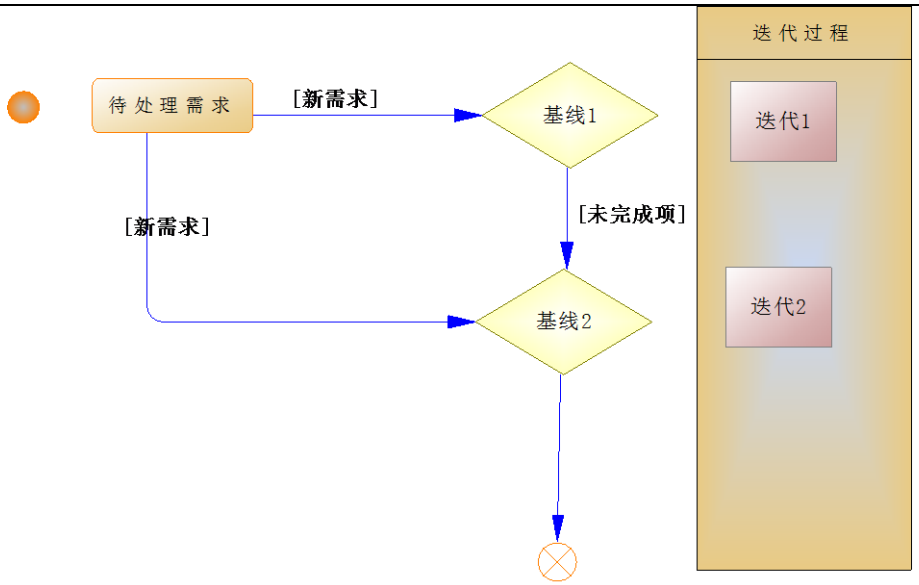
	4、需求分析是一种规格化活动 *2 内容与形式* 分析是任务，建模时手段，利用建模可以用图形代替文本，以更加可视化的方法表示信息，产生的结果并不一定非得是规范性很高的模型 建模的主要要点： 1、尽可能进行团队建模 2、大胆使用草图建模 *3 何时开始与结束* 迭代式需求分析： 1、分析活动逐渐从本质需求过渡到边缘需求 2、RUP 中的细化阶段是需求分析活动最密集的阶段 3、到了 RUP 中的构建阶段，需求分析活动将逐渐减少
3 编写规约	良好的源代码和注释就是最好的文档 软件规格说明书应具有： 1、共享；可获得（软件开发团队可以在开发需要时获得最新版本的规格说明书），可获知（利用开发模板确保软件需求说明书的读者知道自己需要的信息在哪些章节） 2、更新；专人更新（按不同章节指定更新人），写作风格（确保一类信息只在一处描述）

4 需求验证	需求验证的关键手段：评审 主要要点：分层次、分内容进行验证

3.2.3 需求管理工作要点

需求管理项之间的关系



1 统一、明确的需求项划分标准	成功的划分满足以下条件： 1、粒度均匀（如每个需求项的大小相当，即工时相当等） 2、大小合适（如每个需求项工作量以周为单位） 3、完整（最低一级的需求项应该涵盖所有的开发任务）
2 引入基线管理	
	每次迭代都是一个小型的瀑布型生命周期；通过这样的分解，

	整个开发工作就被划分成了多个小项目，这种模式更容易使开发人员保持良好的工作节奏。 在划分基线是，通常需要完成三个方面的事情： 1、确立优先级；确保高优先级，高风险的需求项在尽早的迭代中完成； 2、工作量估算；确保每次迭代的时间安排是紧凑的 3、未完成项的合并
3 引入变更管理	客观存在的需求变更是引入变更管理成为必然 就需求管理而言主要完成： 1、业务影响分析： 从业务角度对变更的合理性、优先级以及对原有需求的影响进行分析，确定优先级 2、技术影响分析： 从技术角度对变更的影响范围、工作量进行分析，并决定是拒绝、在后续还是在本次的迭代中进行响应 3、项目影响分析： 基于前面的工作量分析，考虑是否对整个项目的时间、进度、成本产生较大影响
4 引入需求跟踪	引入需求跟踪能精确地评价在变更的影响分析过程中变更将影响的哪些需求项、哪些设计元素
3.2.4 需求分析人员的技能组成	
1 需求分析人员的来源	1、开发人员；2、用户；3、领域专家

2 各种能力培养的要点	
3.2.5 seru 模型概述	
3.3 小结	

第 2 部分 需求开发	
第 4 章 需求定义最佳实践	
4.1 需求定义任务概述	
4.1.1 需求定义的时机	

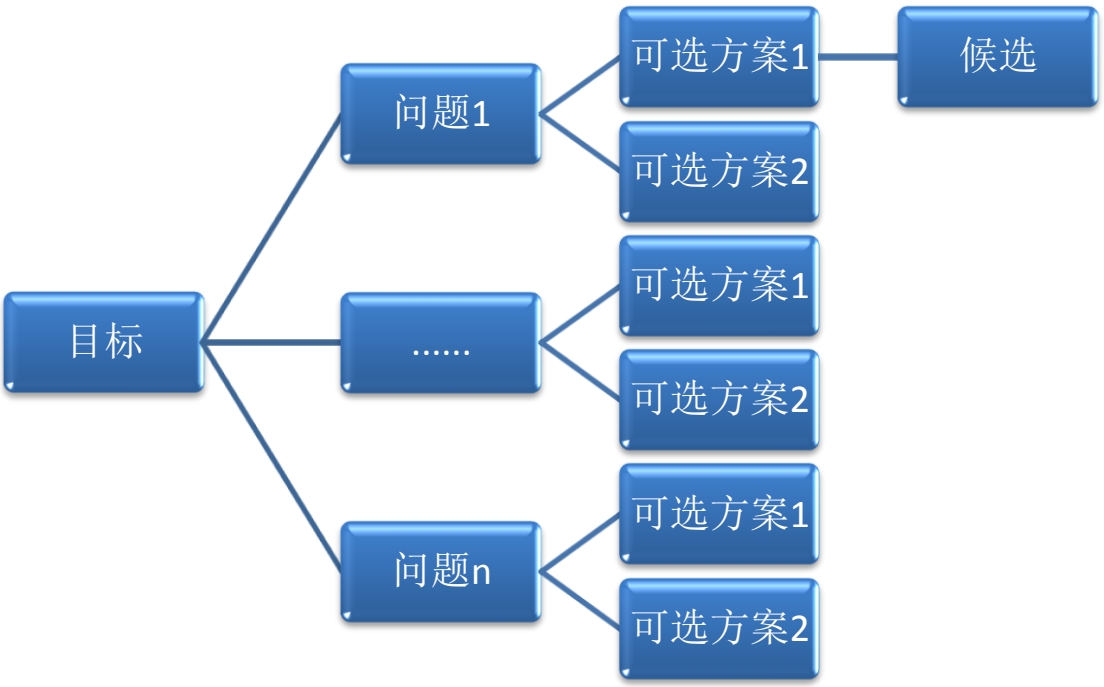
4.1.2 需求定义的理念与策略

1 破解混沌不清的项目目标

解决方案：

1、内部寻根；和项目发起人沟通

2、外部溯源；



2 需求定义的理念

关键：问题和机会

步骤：

目标→问题→可选方案→建议方案

目标：通过内、外部寻根方法将项目要解决的问题罗列出来

问题；针对目标层面的问题进行分析，找到导致该问题产生的根源

可选方案：针对每个问题罗列出可能解决的方案

建议方案：挑选出比较合理的方案

4.2 问题分析的五步法

4.2.1 在问题定义上达成共识

问题定义模板

写作项目	说明
问题	描述存在的问题
影响	该问题影响了谁
结果	该问题给谁产生了什么影响
优点	预期解决方案及优点

问题定义的技巧

1、转换（如棋盘中象棋的移动是一个迭代的过程，同样也可以转换为一个联通图遍历问题）

2、本源

4.2.2 分析问题背后的问题

也就是寻找问题的本源

有以下两种工具：

1、鱼骨图

优点：

使分析人员将问题的原因而不是症状放在首位

提供一种运用智慧解决问题的新方法

直观、简明、易操作

绘制步骤：

1》选择问题

2》头脑风暴

3》确定原因类型；如人，设备、材料、环境、方法或过程

4）分配原因

5》分析根本原因

6》小结

2、帕累托分析

少数的失误，应该为大量的质量成本复杂

4.2.3 确定相关人员和用户

1、Stakeholders ：筹码持有人

筹码有大到小的顺序：高层-》中层-》操作员

筹码越高说明项目健康度越高，也就越有可能成功

2、用户分析

4.2.4 定义解决方案的界限

1、范围与边界

范围是指系统涉及哪些内容

边界是系统与人的职责边界

2、确定边界

3、边界谈判

4、创新边界

4.2.5 确定加在解决方案上的约束

1、技术开发

技术约束（如平台）

预期软硬件环境（兼容性）

预期的使用环境（如户外，或特殊的作业环境）

2、项目实施

行政约束（如许可）

进度及资源约束（如进度要求）

环境约束（如合法）

4.2.6 小结	

4.3 需求定义的产物与要素

4.3.1 需求定义的产物

1、项目综述（POS）

主要内容：

目标—业务目标

相关人员和用户 - 与系统交互的人

限制条件—必须采用某设计方案，时间，经费等

关键术语

相关事实与假定---项目的提出背景，技术能力

工作范围 - 系统设计的内容范围

费用计划

风险—面临的主要风险

可行性 -包括技术、经济、社会可行性论证

2、愿景Vision

主要内容

内容类别	内容项	说明	POS对应项
业务需求	背景 业务机遇 业务目标 客户需求 提供给客户的价值 业务风险		相关事实 目标 风险
解决方案	愿景说明 主要特征 假设与依赖		 工作范围 相关事实
范围和局限性	首次发行范围 随后发行范围 局限性		工作范围

业务环境	客户概貌 项目优先级 操作环境		人员用户
产品成功因素		成功的标准及定义	费用计划

4.3.2 需求定义的要素

1、目标

满足:

1>具体　　2）可度量　　3）可达到

4）和其他目标具有相关性　　5）必须具有明确的截止期限

4.4 定义需求范围

4.4.1 案例说明	

4.4.2 划分主题域

1、主题域的定义

划分主题域的原因:

由于利用子系统的划分方法,在进行需求捕获和分析是就会发现各个子系统和模块与客户部门是交错在一起的,每个模块都需要对不同的部门进行调研。它只是一种逻辑划分，并没有很好地把业务结构体现出来，它只是采用“业务名字+管理”的形式命名的，其中业务名词实际上就是业务实体，也就是物的线索；对于大多数业务系统而言都不是最佳的分解方式，因为这些业务实体会牵涉到大量的业务流程。

合适的划分是根据业务流程，也就是以“事”为线索

2、为什么使用构件图

使用构件图与系统的层次结构图只能体现系统的组成关系相比,更能体香每个主题

域之间的关系

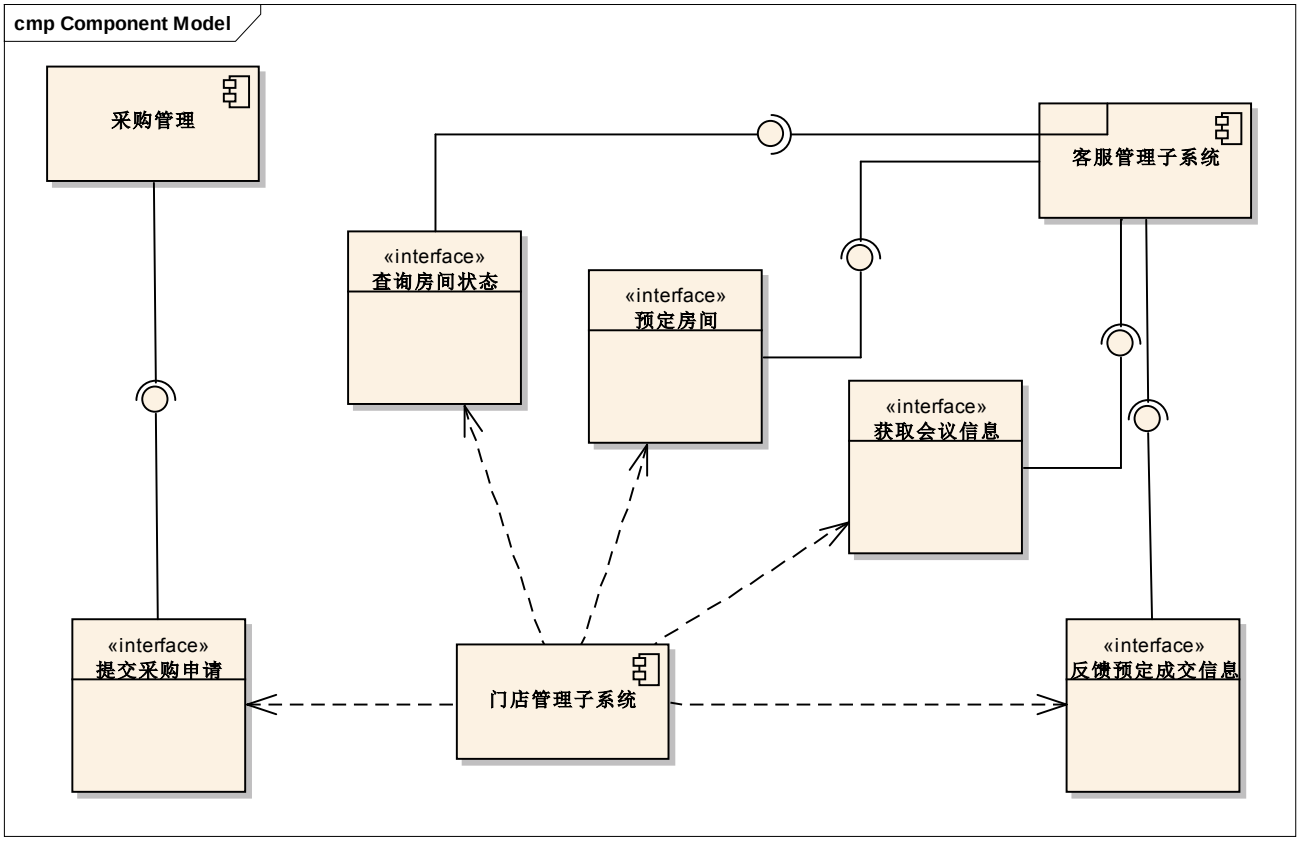
1》 服务接口的重要性

尽早的标示出各个主题域之间的接口，可以有效地避免后续系统对当前系统的影响

2》构件图解析

3》构件

4》服务接口：利用职责驱动设计的思想设计接口



4.4.3 确定主题域范围

1、上下文关系图绘制的要点

通过绘制上下文关系图来确定主题域的范围

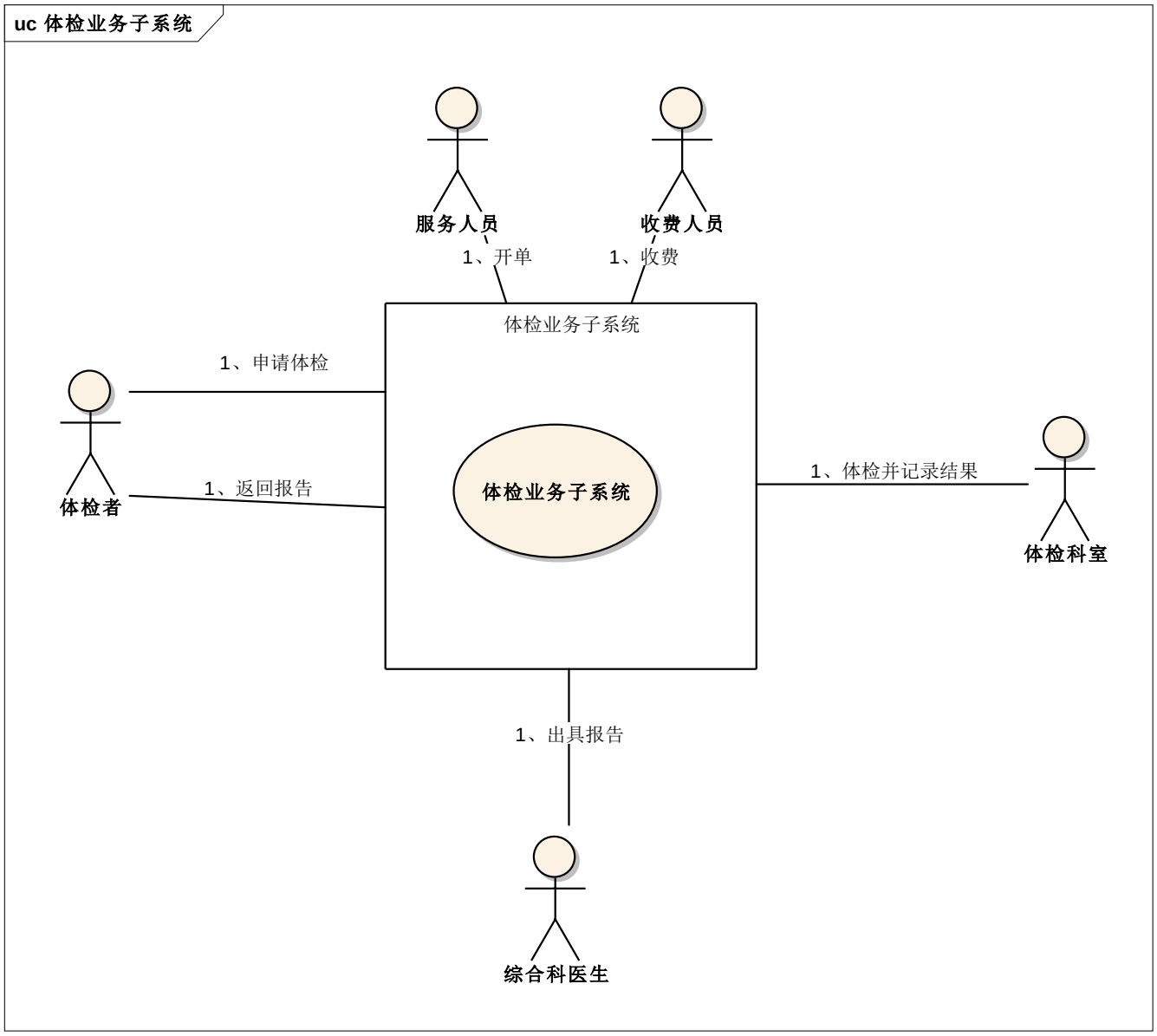
步骤：

1、首先画一个矩阵，写上系统的名称

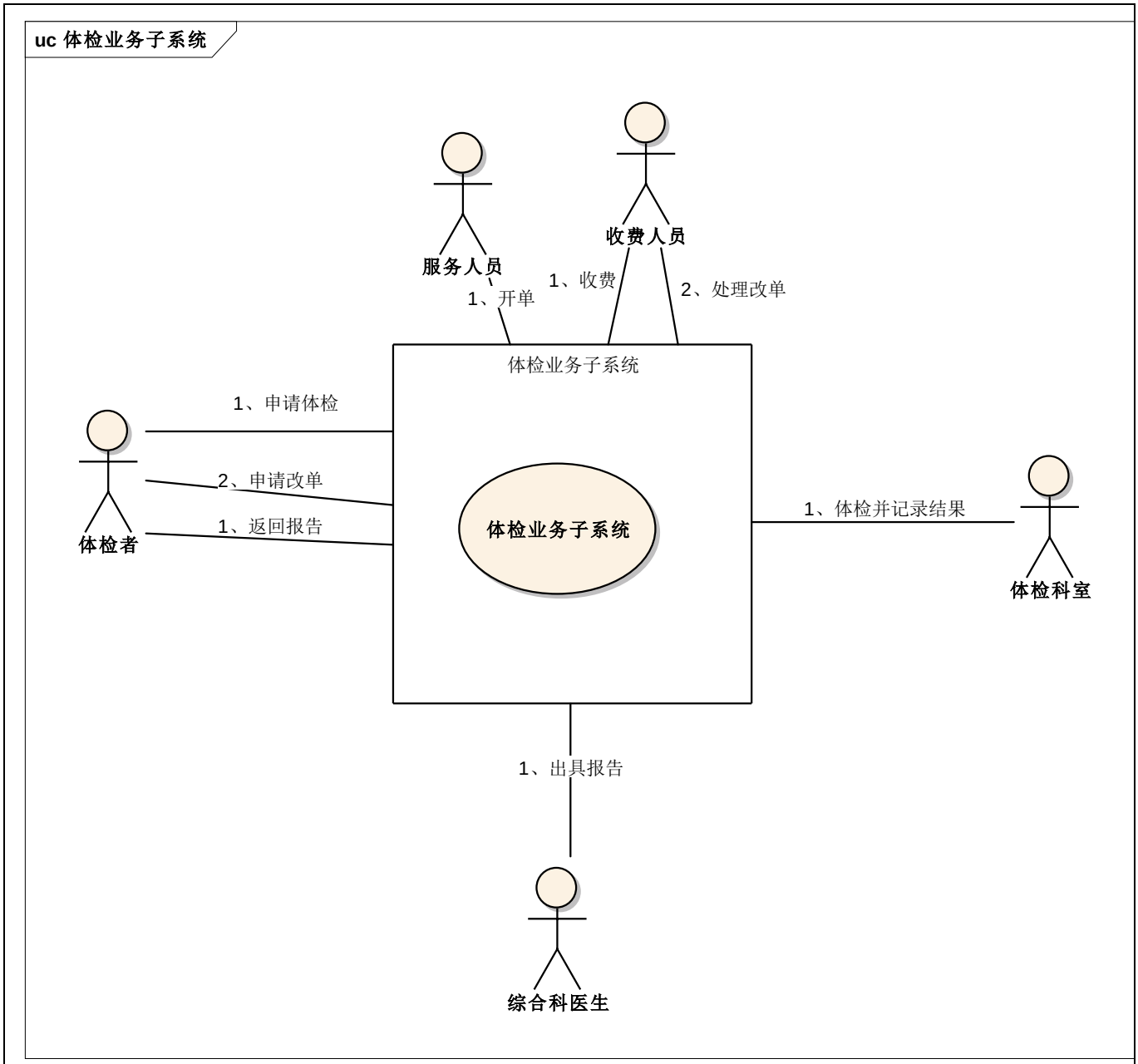
2、找到该系统的所有客户，考虑这些客户会发起什么事件，这些事件会引发内部工作人员的什么工作，将这些序列列出

3、最后看看每个内部工作人员有没有一些主动发起的事件

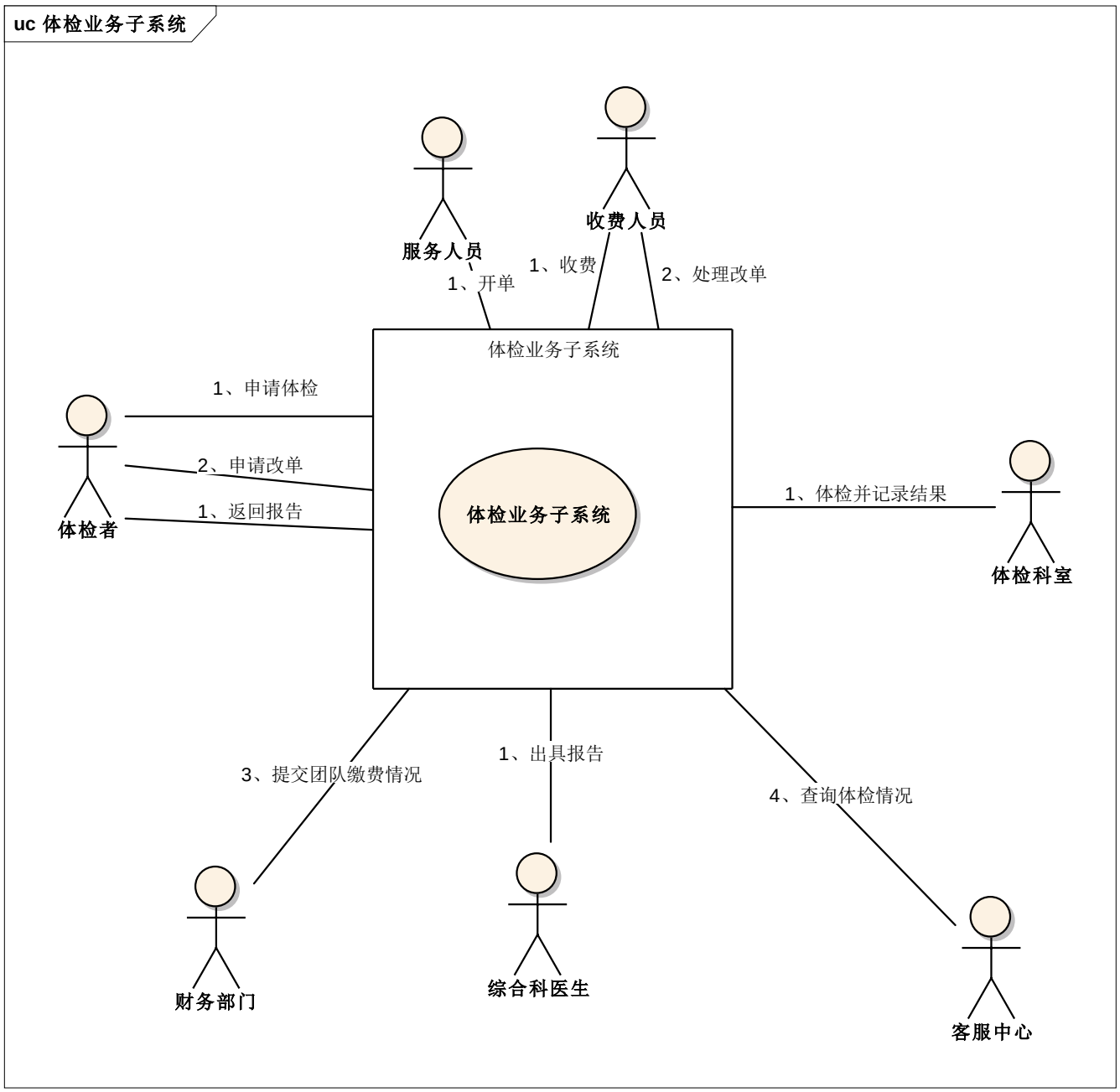
第一步：对主题域有哪些外部用户和内部要做些什么



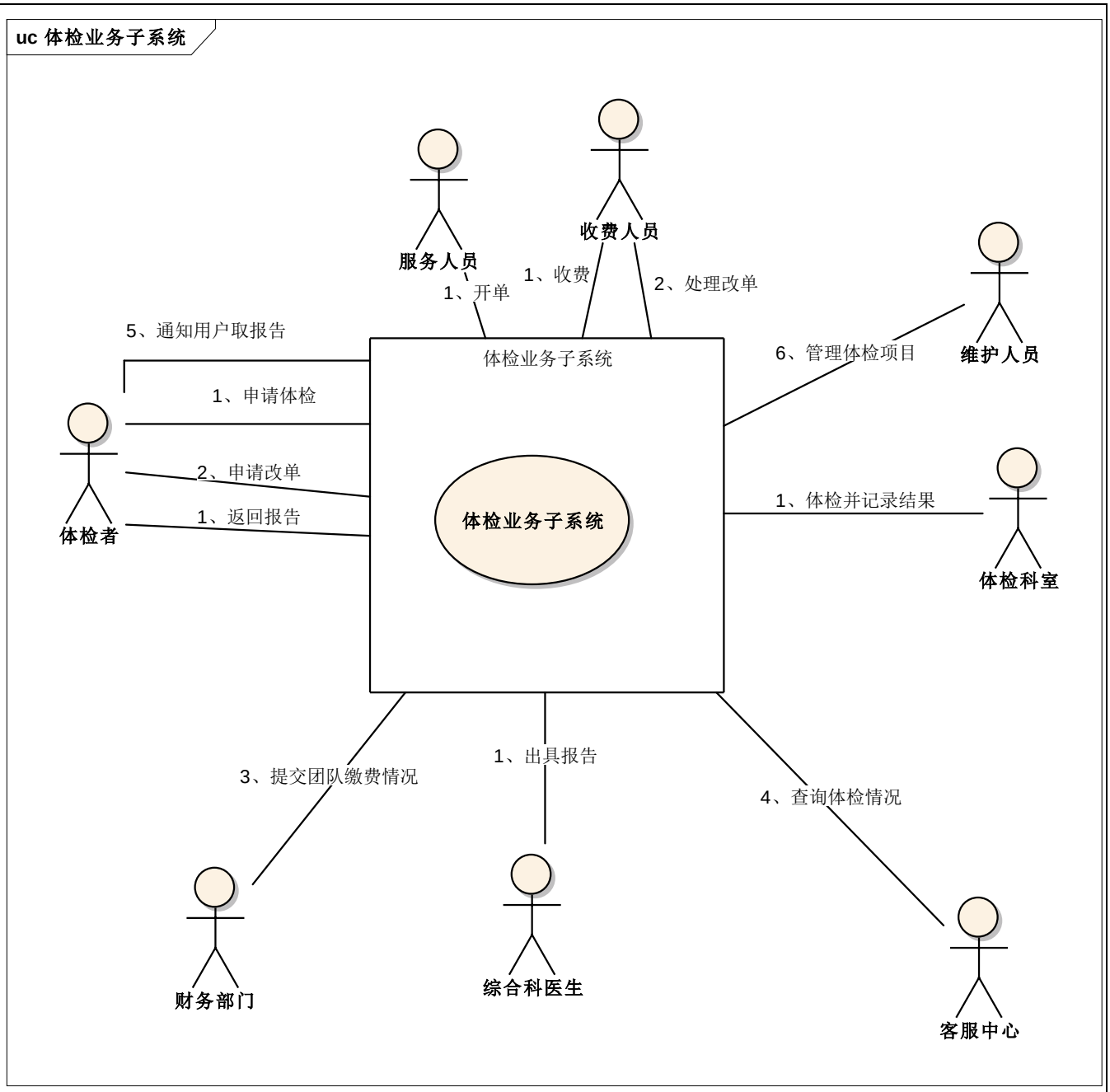
第二步：体检者除申请体检之外还有什么独立行为：如修改体检项



第三步：寻找是否有除系统主要发起人之外的用户发起的事件；如财务部门



第四步：考虑内部的 Worker 是否有主动的行为。



4.4.4 标识业务事件与报表

1 业务事件解析	业务事件是业务流程的触发点，标示出业务事件能够帮助我们认识业务流程，而业务流程是为了响应业务事件而触发的一系列业务活动，它通常是由于不同部门、不同岗位协作完成的，业务流程的信息掌握在中层管理人员的手里，它属于脉络信息。

	业务活动是一个特定的业务流程，它是一个人的活动，因此一个业务流程应该是有一个或多个业务活动组成的：而业务步骤是完成某个业务活动所需要的具体步骤。因此业务活动和业务步骤的信息是掌握在操作层人员手里，它属于细节信息。
2 业务事件标示要点	1、业务事件类型 业务事件可以分为外部事件（系统参与者发起）和内部事件（如状态事件和时间事件） 2、业务事件标识要点 业务事件是主动触发的，直接作用于系统的，也就是将触发系统行为，并且会产生一系列后续行为 如： 客户想买一件衬衫（业务事件）—》 客户开车到购物中心---》 客户在 xx 点试穿衬衫---》 客户走进我们的店铺---》 客户在我们店铺试穿衬衫----》 客户购买一件衬衫； 注意事项：在梳理业务是应该先把诸如数据备份、更改密码之类的技术性活动放在一边，因为我们梳理的是业务事件，而不是系统事件
2 案例分析	

4.4.5 生成需求大纲	
4.5 小结	

第 5 章 需求捕获最佳实践	
5.1 需求捕获的策略	
5.1.1 需求捕获应该是主动的	需求捕获是撒网打渔，而不是休闲钓鱼
5.1.2 需求捕获应该是聚焦的	捕获过程中的问题： 1、提不出需求 2、提的需求太多 解决方案： 提出聚焦的问题，产生镀金需求
5.1.3 破解需求的冰山模型	1、意识到的需求： （用户层面） 位于海平面之上，用户自己能够设想到的需求，（大量的需求以变更的形式出现） 2、无意识的需求 （开发者层面） 实际的工作场景，开发者能够对场景“感同身受”的话，那么就可以大大减少变更的数量，需加强业务知识的捕获

	3、未梦想到的需求； （系统架构师层面，找到最佳的技术解决方案） 用户对技术解决方案永远都不是擅长的，因此他们无法构想出对其工作产生革新性变化的解决方案。这就需要通过需求分析人员在对问题域充分利用的基础上，选择合适的技术方案，才可能创造出用户未梦想到的功能，成为优秀的需求分析人员 即：善于利用技术为用户创造全新体验是优秀需求人员的特质
5.1.4 破解阻碍需求捕获的心理现象	1、言过其实心理 解决方案 1）用户言过其实时的表现：表述都非常肯定的语气，十分流畅。因为此时他只需创造，而不需要回忆。 2）验证；因为在没有串供的情况下，天下谎言皆不同 验证有谎言存在时，解决方案 1- 差异展现法： 把不同用户的不同表述展示给中层后，找共识 2- 瓶颈分析法

<table>
<tr><td></td><td>对流程执行过程中的瓶颈进行分析（如都需经理审批），避免流程瓶颈导致系统无法顺利运转
2、越俎代庖心理
识别正确的被访谈者最为关键
1）问题的层次是否正确
高层—宏观、中层—脉络问题、操作---细节
2）根据业务背景判断
3、非正事心理
解决方案：对访谈进行计划
4、抗拒心理
激将法，倾听对方的抱怨是化敌为友的有效手段</td></tr>
<tr><td colspan="2">5.1.5 不要忽视对变更可能的捕获
针对常见变更类型的类型
<table><tr><td>变更类型</td><td>捕获问题</td></tr><tr><td>流程变化快</td><td>1、你们上次组织结构调整是什么时候？
2、这个流程以前也是这样的吗？如果不一样，是什么时候的事呢？</td></tr></table></td></tr>
</table>

业务规则变化快	1、这些规则是来自外部法规，还是内部制定的？ 2、如果是外部法规，这些法规通常几年会更新一次？ 3、如果是内部制定的，那么是由哪个部门制定的？依据什么进行调整？
用户界面变化大	1、用户界面设计一般由谁确定？ 2、你们最满意哪个系统的用户界面？ 3、你们最不喜欢哪个系统的用户界面？
5.1.6 需求协商	1、揭开解决方案后面的问题 2、共赢性谈判 共赢的谈判就是抛开立场,追求满足大家的利用诉求 3、转换技巧 1 --） 相对重要→ 相对次要 2 --） 关注点转换 3---） 隐喻
5.2 需求捕获的主要方法	

5.2.1 用户访谈

考虑因素：

1、优缺点与使用时机

2、用户访谈的类型

3、时间安排

开场白 - 陈述预先的理解

预先计划问题 -- 寻求问题的答案

即兴问题 ---扩大需求信息量

总结 - 总结访谈内容

4、用户访谈中的记录工作

5、用户访谈中的沟通技巧

1—） 制作访谈问卷并事先发给被访谈者

2--） 把握语言节奏

3--） 有效结合不同的问题类型

开放式问题

封闭式和半封闭式问题

4--）善于安排问题顺序

金字塔结构：一种归纳过程

漏斗结构：一个演绎的过程

菱形结构；上述两个结构的混合

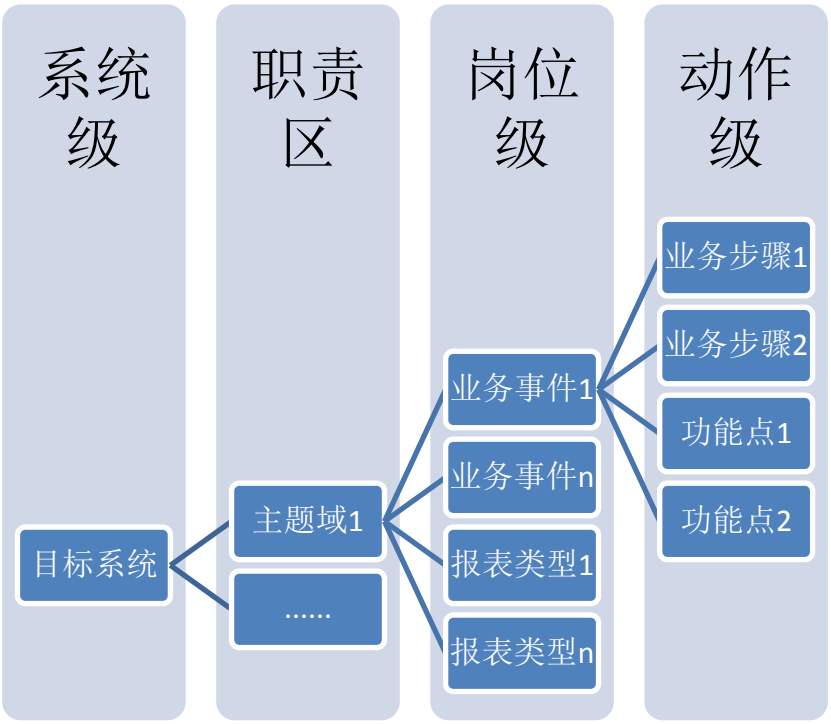
5--） 注意沟通的细节

让模型成为访谈过程中的工具

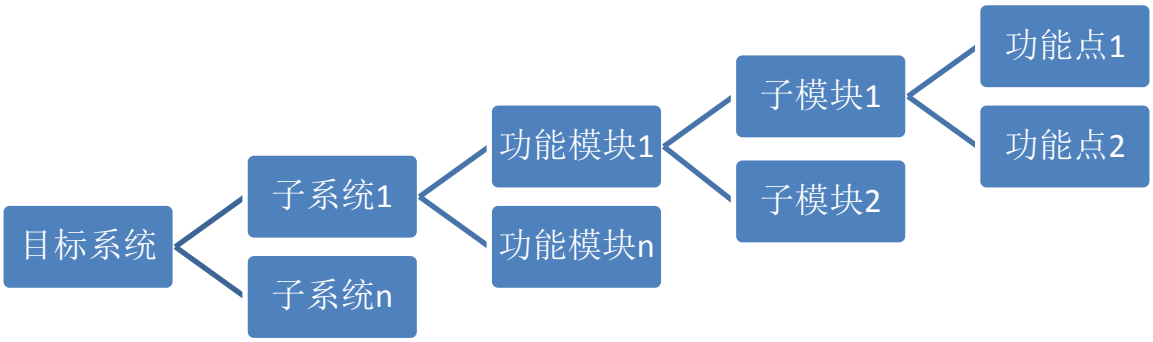
避免出现一些干扰访谈的暗示

5.3 需求捕获的记录工具	
5.3.1 工具的选择与定义	沟通决定内容，内容决定格式
5.3.2 任务卡片	任务卡片的内容包括 任务　　目的　　触发 前提　　频率　　关键情况 子任务（如具体业务步骤） 任务变体（如异常）
5.3.3 场景说明	
5.3.4 其他工具	
5.4 小结	

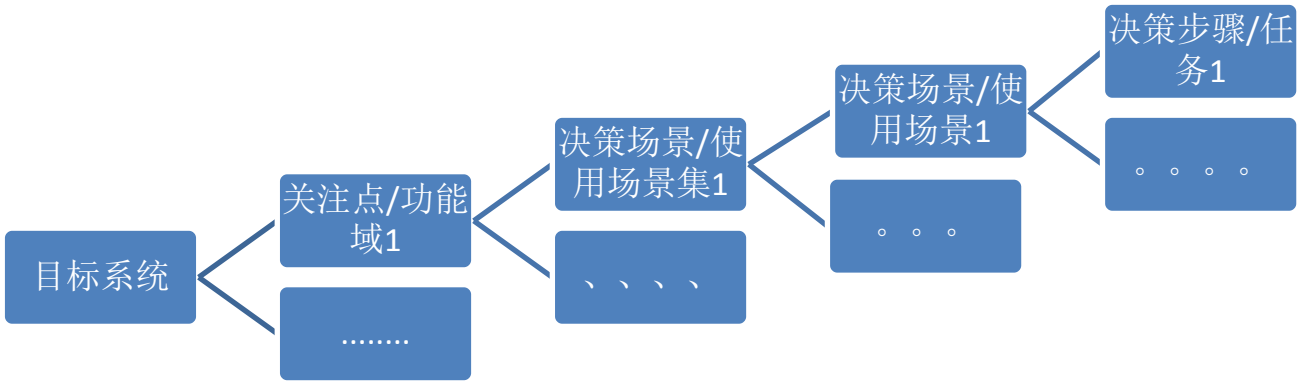
第 6 章 需求分析与建模最佳实践
6.1 需求分析与建模的要点与误区分析
6.1.1 需求分析到底做什么
1、分解 按业务流程为主线索德分解结构（按事的角度分解）



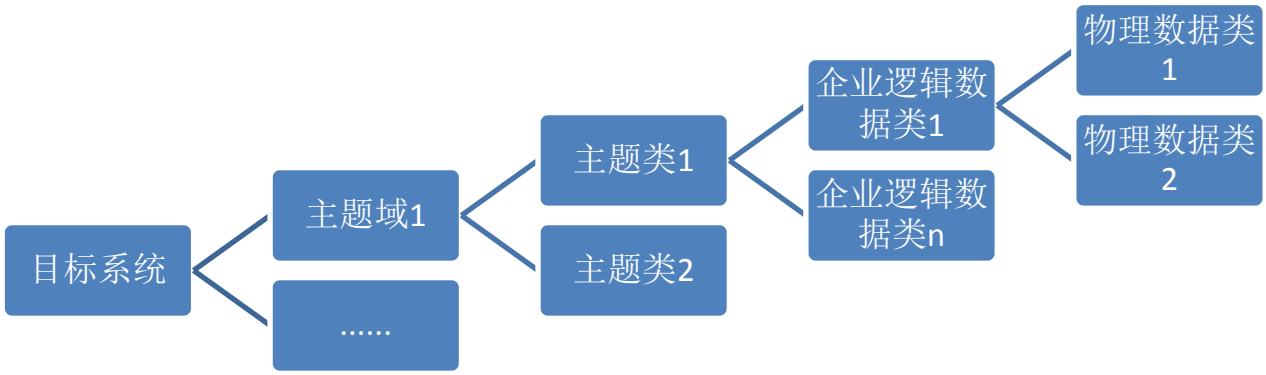
（2）按程序结构为主线索的分解结构



（3）基于场景的分解结构



（4）基于数据分解的结构（按物分解结构）



2、提炼

3、消除矛盾

6.1.2 建模的目标与要点	
6.1.3 选择建模工具的要点	
6.2 周期一：理清框架与脉络	
6.2.1 业务流程分析	

1、业务流程分析任务概述

业务事件经分析识别现有业务活动，确定业务活动之间的关系，了解业务活动需要接受那些信息，将产生那些数据，确定数据的传送路线，同时标示与那些部门相关

2、业务流程分析与流程管理理论的关系

业务流程具有内在性，有利于可分业务之间的耦合

（1）流程的六大特性

1、目标性　　2、内在性　　3、 整体性　4、动态性　5、层次性　6、结构性

（2）工作流实现的本质

备注：根据业务的耦合性划分相应的模块

（3）流程设计的原则

1、以产品为中心，而非任务为中心

2、让那些需要得到流程产出的人自己执行流程

3、在决策点位于工作执行的地方，在业务流程汇总建立控制程序

4、流程改进的 ESIA 策略

3 业务流程分析的要点与产物

（1）流程是有层次的（组织，部门，岗位）

（2）流程是有类型（生产性、管理性、支持性流程）

（3）流程分析的产物

跨职责流程图

主要元素：流程的名称、阶段、元素、并行、引用

职责带按事件发生的顺序从左到有排列（各个部门、组织、及个人）

完成的图应具有的特征：

1、职责带上的命名都将细化到具体的岗位；

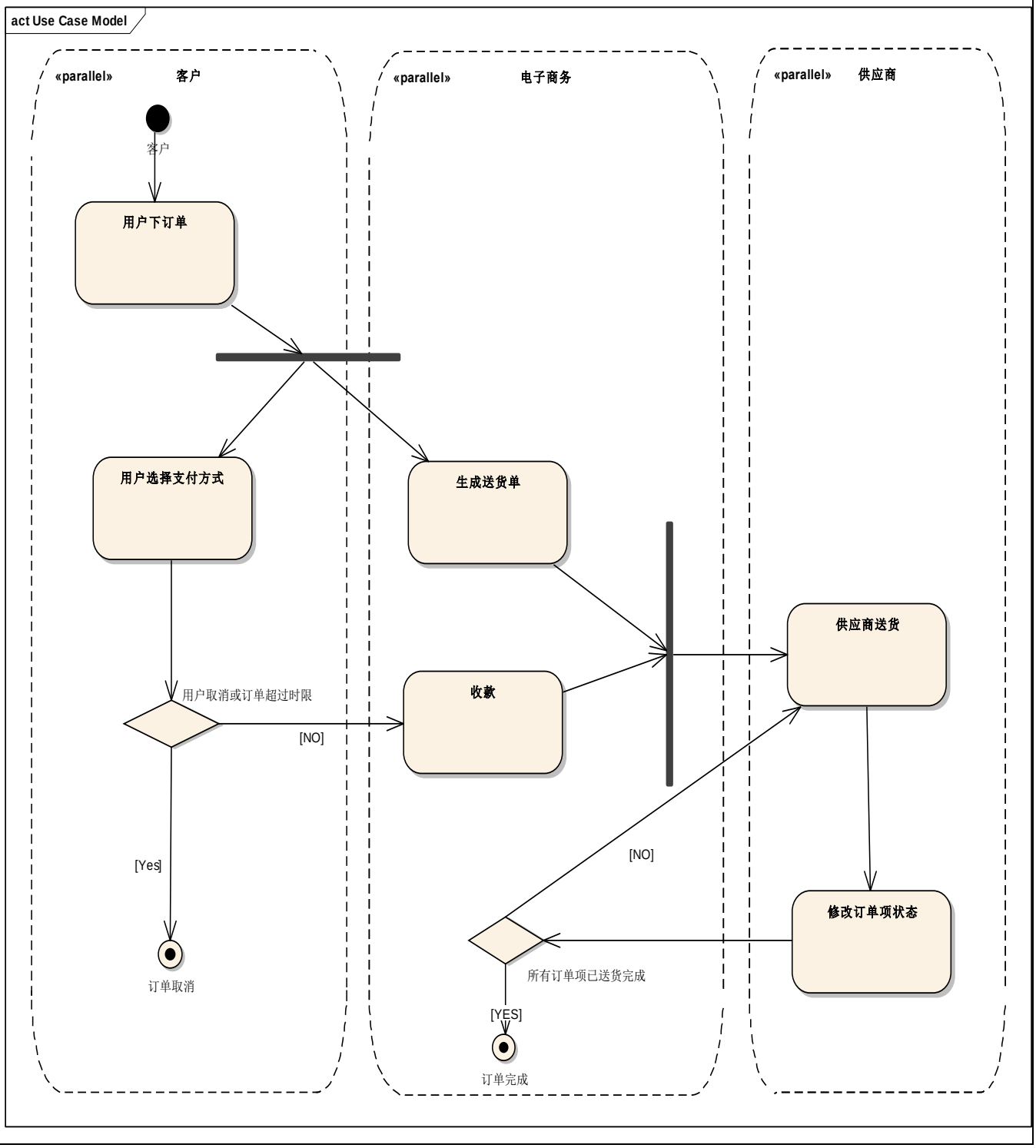
2、每类活动之间的关系已经没有疑问，大家都达成了共识

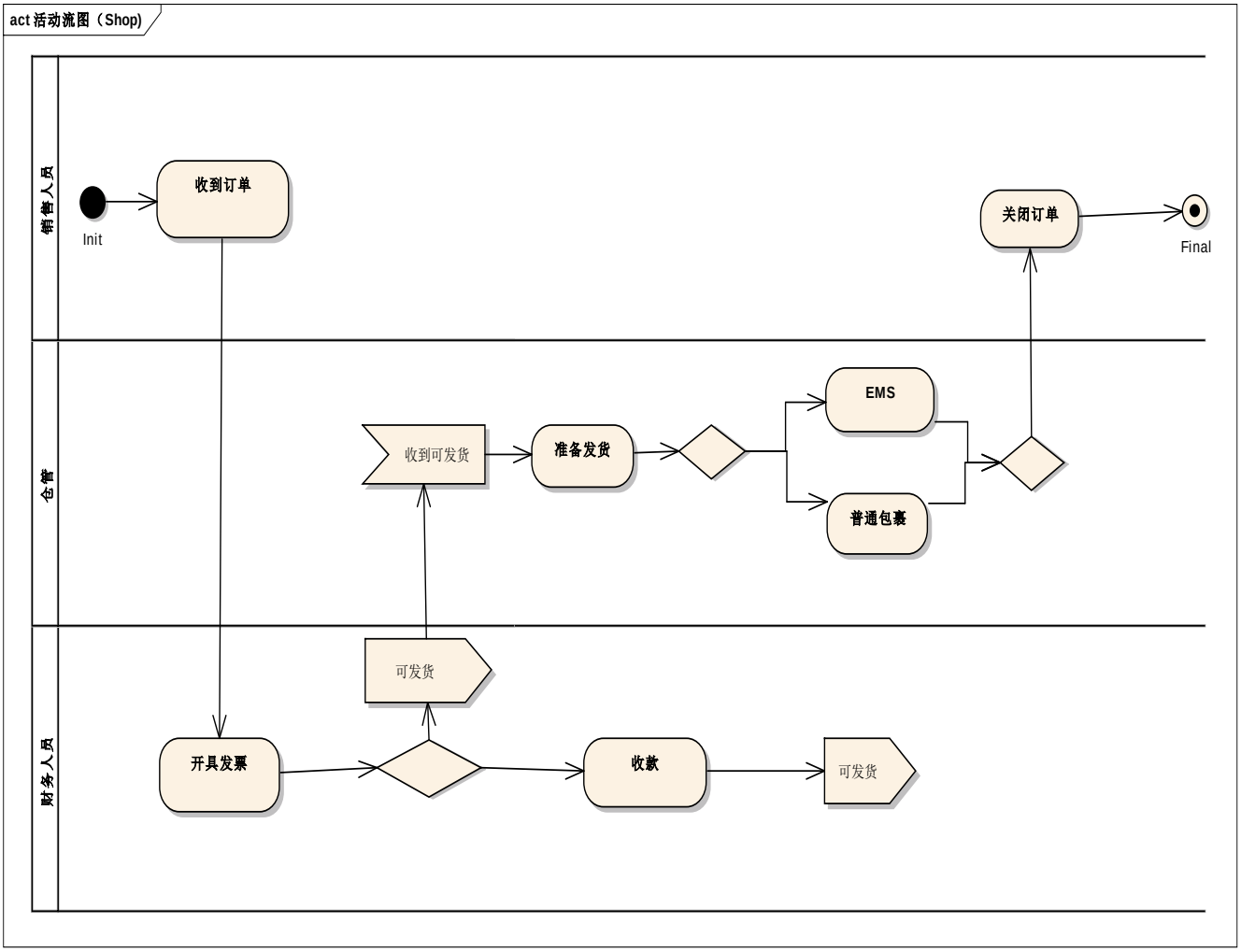
主注意事项：不要对业务活动进行细化

活动图

活动图与流程图的区别：

活动图具有并发性，流程图没有





数据流图

它对于数据流为主线索的处理过程是最适合的，例如计费系统。当然，如果你要将其应用在业务活动的描述中也是可行的，只不过你必须忍受无法表示

分层数据流图

引入层次结构的数据流图，它是按照系统的层次结构进行逐级分解的，以分层的数据流图来反映这种结构关系。

1、构建顶层图

2、根据业务事件绘制 DFD 片段

3、将 DFD 片段合并成 DFD

6.2.2 业务实体分析

领域模型：它是以面向对象的视角看待现实世界的结果，也就是通过类图来描述现实世界中各种事物之间的关系。

构建的过程主要是找出相关的类，然后命名类之间的关联关系，必要时加入一些多重性描述和业务规则约束。

分析模型：在用例模型和领域模型的基础上进行综合分析而得到的

分析模型中三种十分有用的构造型

实体类：实体对象的抽象，通常来自域模型也就是现实世界，用来描述具体的实体，通常映射到数据库表格与文件中

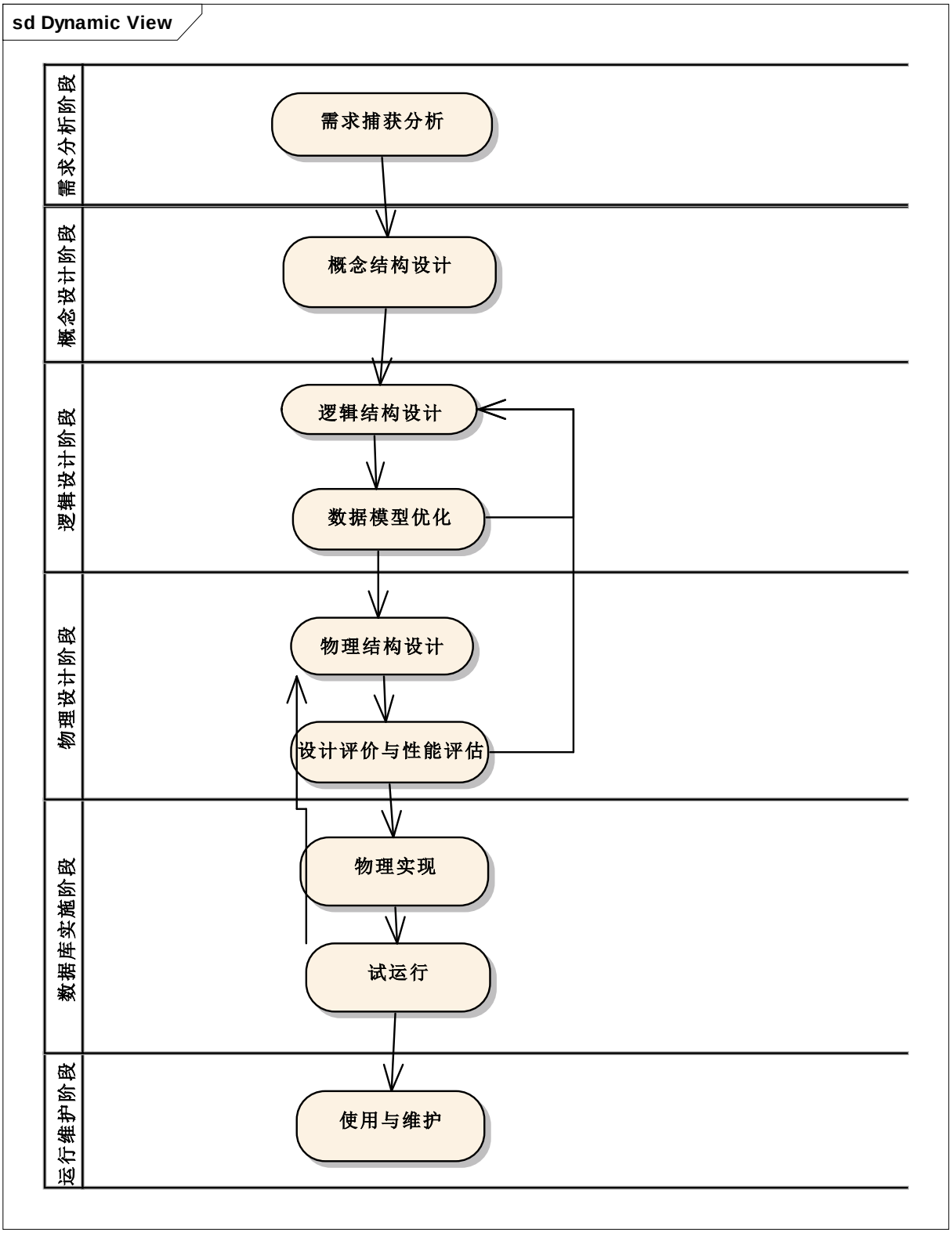
控制类：控制对象的抽象，主要用来体现应用程序的执行逻辑，将其抽象出来，可以使得变化不影响用户界面和数据库中的表。

边界类：边界对象的抽象，通常完成参与者（用户、外部系统）系统之间交互的对象。如对话框、菜单

设计模型：在分析模型的基础上添加设计元素的结果。与分析模型相比，设计模型中类的属性集更趋完善；更重要的是，它将加入模板类、参数类、抽象类/接口等设计元素，以及框架类的使用、设计模式的使用等。

3、E/R 图应用基础

（1）数据建模过程



完整性：逻辑模型在完整性上要比概念模型更胜一筹，通过在需求细化、设计阶段会对垒的属性进行细化，会补充一些新的类。

加工方式：概念模型的原则是忠于问题域，而逻辑模型则会从实现的便利性和需要

的角度进行细化，具体来说可能会对一些类进行分析、合并。

E/R 图应用基础

E/R 模型的主要元素

1、基本元素

实体：也就是问题域中的业务实体，它等价于“类”，是对一类业务对象的抽象，是 E/R 模型中主体的元素。

一种是与类图相似的矩形，另一种是带圆角的矩形。

属性：也就是描述业务实体的字段信息

关键属性：也就是键值，如主键和外键等信息。

关系：当描述两个实体之间的关系时需要指定一些特殊的属性，例如任职关系（需要指定任职时间、职位信息）、借阅关系。这是可以将这些信息打包成一个“关系表”，它等价于“关联类”。

2、元素之间的关系

在 E/R 模型中，只能够建模关联关系。它可以区分可选关系、强制关系以及数量关系。

6.2.3 角色与使用场景分析

1、用例分析技术概述

用例分析的内容包括：

（1）用例图作为目录　　（2）用例描述是封装所有需求的形式

2、参与者用例

包含：参与者、用例

定义

参与者：在系统之外，通过系统边界与系统进行有意义交互的任何事物。

判断的要点：他们是不是系统行为的触发者

它是一种角色，而不是一个特定的人

参与者是直接使用系统的最终用户

系统边界：它是逻辑概念上的一种职责边界而非物理边界。

简单说就是“待开发系统”

用例： 用例实例是在系统中执行的一系列动作，这些动作将生成特定执行者可见的价值结果。一个用例定义一组用例实例。

用例场景是又步骤的，它是一个由一系列业务步骤组成的业务活动

用例场景是有目标的，它能够为参与者带来有意义的结果

用例是对一组用例实例的抽象，用例是有路径（基本事件流、扩展事件流、子事件流）

用例和具体场景之间的关系和类与对象之间的关系是类似的，一个场景是一个具体的行为，一个用例是对一类相关行为的抽象。

箭头的方向说明：

被指向的用例属于源用例

（1）包含关系

表明基用例在它内部说明的某一个位置上显式地合并了另一个用例的行为。

用例粒度大小说明：

只有当某个事件流片段在多个用例中出现时，我们才将这个事件流片段抽取出来，放在一个单独的用例中，这样就可以简化基用例的事件流描述，同时也使得整个系统的描述更加的清晰。

用例粒度控制的要点：

1、业务价值判断是关键

每个用例都有价值，而包含用例和扩展用例是子事件流和扩展事件流

2、影响用例大小的关键是业务流程，是工作任务的分工

当系统动作和业务名词在用例名称中相遇时，就是一个非常危险的信号。

（2）扩展关系

它表示基用例在由扩展用例间接说明的一个位置上,隐式地合并了另一个用例的行为。

扩展用例只在特定的条件下，它的行为可以被另一个用例的行为所扩展

（3）泛化关系

表示子用例继承了父用例的行为和含义；子用例还可以增加或覆盖父用例的行为；子用例可以出现在父用例出现的任何位置。

注意要点：千万不要为了使用扩展、包含关系而使用它们。扩展用例建模的通常是使优先级较低的扩展事件流,包含关系建模的通常是多个用例所包含的公共子事件流。

（4）参与者之间的关系

只有一种：泛化，作用是降低模型的复杂度

泛化的是角色之间的共性

4、用例的来源

方法：自顶向下导出法

从流程图中获取

步骤：

1、 边界确定（去除非直接使用系统的岗位）

2、确定角色（对剩下的职责带区进行角色化）

3、确定用例

自底向上合并法

用例图一般用先事（场景、用例），后人（角色、参与者）

也就是先将事找到，后将人对应到事上，从而画出用例图。

6.2.4 周期一的产物

1、工作任务说明

需求分析的第一阶段，核心任务是结合业务流程、报表的需求，梳理出结构框架（领域模型）和行为模型（流程图➡用例模型），为第二阶段的需求分析工作建立基础，指出方向。

2、业务事件分析

一、业务流程分析

对不影响泳道间协作的判断、活动均属于细节信息，不用显示在活动图中。

二、 业务实体分析

关键是理清问题域中的关键术语之间的关系，我们可以针对每次流程分析的过程中获得的信息，标示出来，确定他们之间的关系，并用类图表示出来

6.3 周期二：确定需求细节

6.3.1 确定行为需求的细节

1、用例的灵活应用

业务功能、报表功能、接口、技术支撑

用例描述模版

Table 6.0	
Use Case #	DATAENTRYPROJECTCUST-1009
Use Case name	Maintain Customer
Description	This Use Case depicts full maintenance of customer from project "Data Entry".
Scope and level	• Data Entry System (Internal) • Credit Card System (External)
Level	User Goal Level (If this property is not understood, look at the reference for the book Writing Effective Use Cases (**PRE-PUB. DRAFT#3**): Alistair Cockburn Humans and technology)
Primary and secondary actors	Data Entry operator.
Stakeholders and interests	
Trigger	Data entry operator clicks on menu: "Add New Customer"
Preconditions	• Data entry operator should be logged in. • Data entry operator should have access to Internet.
Assumptions	Customer information received is entered manually. No automated import routine is in the scope.
Failed End condition	• Customer is not added to database and appropriate error message is displayed. • Customer code already existing in the customer database. • Customer code length limit is exceeded. • Customer credit card limit is exceeded. • Customer credit card validation failed with the payment gateway.
Action	Add new customer

Main success scenario (or basic Flow):	1. Data entry operator receives customer information.
	2. Data entry operator enters following information:
	  o Customer code
	  o Customer name

	○ Customer address ○ Customer phone 3. Customer code is checked if it exists in Customer table. ○ If the customer code is existing then "Duplicate Customer Code" error is raised. ○ If the customer code is more than 8 length, then "Customer code length limit crossed" error is raised. 4. After step 3 is passed OK. Data entry operator enters credit card information. If the credit card length is more than 10 length, then "Credit card length limit crossed" error is raised. 5. Credit card information is send to the external payment gateway. Appropriate APIs of the external payment gateway will be used for validity. 6. External payment gateway returns "OK" if credit card is validated or else will return "NOT VALID" flag. 7. Data entry operator then adds the customer in database.
Alternate scenario (Extensions):	Update Existing Customer 1. Data entry operator enters customer code to retrieve the customer who has to be updated. 2. Data entry operator makes appropriate changes to the customer information. All steps and business validation from 1 to 6 of Add new Customer is repeated. 3. Data Entry operator updates the customer information.
Alternate scenario (Extensions):	Delete Existing Customer 1. Data entry operator enters customer code to retrieve the customer who has to be deleted. 2. Data entry operator deletes the customer. Data entry operator is alerted "Are you sure you want to delete the Customer?" ○ If the data entry operator clicks "Yes", then the customer is deleted from the database. ○ If the data entry operator clicks "NO", no action is taken.
Success Guarantee (Post conditions):	• Customer is added to Customer database. • Customer is updated to Customer database. • Customer is deleted from Customer database.
Special Requirements (including business rules):	
Technology and Data Variations List:	If credit card payment gateway API changes, the interaction of the data entry customer module will have to be changed accordingly.

Frequency of occurrence:	
Notes and Open Issues:	

第 8 章 需求验证最佳实践	
8.1 需求验证的主要手段	
8.1.1 不同正式化程度的评审	
8.1.2 审查过程概述	
8.2 需求验证的主要误区与解决方案	
8.2.1 需求验证的 5 大要点	
8.2.2 需求验证常见的 5 大问题	
8.3 小结	

第 3 部分 需求管理	
第 9 章 需求基线操作实务	
9.1 需求基线的理念与策略	
9.1.1 基线思想的起源	
9.1.2 基线的策略	
9.2 基线划定的基础：优先级评价	
9.2.1 组织需求项	
9.2.2 业务优先级评价	
9.2.3 根据技术依赖性和项目风险调整优先级	技术上、管理上风险越大越提前开发，提高它的优先级
9.3 基线划定的要素：工作量估算	

9.3.1 估算的意义与要点

估算是一种手段，其目标是追求管理上的可控性，而非估算结果的准确性

软件估算类似于财务估算活动，估算的结果是一个可控范围，需动态调整，由各个自项的值累加起来的

1、何时进行估算

在需求定义阶段完成时，应安排一次估算

需求分析一阶段完成时，也应安排一次估算

每次迭代开发完成之后应该填充实际进度数据，并调整估算值

2、估算的核心思想

寻找计数单元，考虑复杂因子（如：外部查询、外部接口文件、外部输入、外部输出、内部逻辑文件皆为计数单元的复杂因子，难以估算）

计数单元的寻找方法：

（1）需求定义阶段；可以考虑业务事件、报表类型、接口为计数单元

（2）需求分析一阶段：用例可以作为计数单元

（3）需求分析后阶段；可以考虑使用常用功能点（FP）或 COCOMO 中推荐的计数单元

3、建议的估算操作要点

一、分部分、分内容 ；二、采用权重

9.3.2 定义阶段的估算示例

1、对每个主题域进行估算

业务流类

（1）采用 Delphi 法作为一种常用的专家评估技术

（2）其他类 后续估算任务
9.3.3 分析一阶段的估算示例
9.4 基线划定与管理
9.4.1 划定基线 当确定了所有需求项的优先级、对其工作量做了基本的估算之后，就可以开始划定基线了，因此我们采用的是“早基线，晚冻结“的策略 原则： 优先级越高，安排在越早的基线中（关键、重要、有用、一般）
9.4.2 管理基线 对预先的“划定“的基线产生影响： 1、需求变更；将其更新到“待处理需求”集中，不影响原基线的划定 2、迭代过程未完成；划定两次基线。 一个基线投入开发迭代，另一个基线开始细化需求，从而实现流水线作业
9.5 小结

第 10 章 变更管理操作实务
10.1 变更管理的理念 变更的来源： 1、我们捕获信息的不全面； 2、商业社会变化频繁

需求变更管理的目标是：控制变更，而非避免变更，控制变更的目标是减少变更对开发工作的影响

采用的方案：

1、统一渠道（即集中处理）

2、统一平台（即统一认识）

需求团队的贡献在于“尽早标示变更”

设计团队的贡献在于“尽可能以弹性的设计减少变更的影响”

10.2 变更管理要点一：统一渠道

10.2.1 ccb 背后的道理（变更管理委员会）

CCB 解决的问题是：

1、通过统一规划和管理，控制程序员响应的变更之间存在的冲突

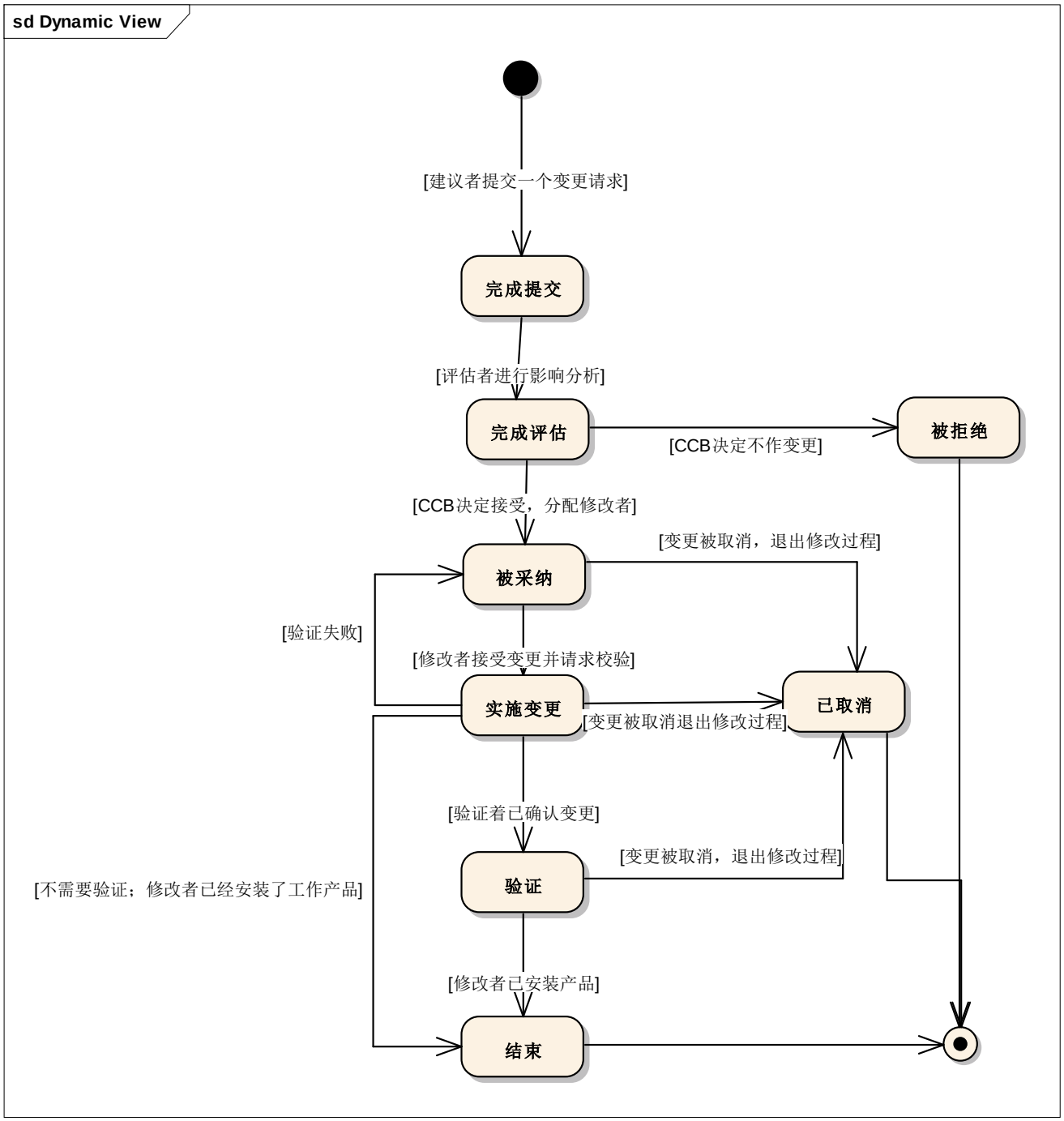
2、通过让客户为变更付费，到达变更量的重视

核心人员（用户团队、开发团队）

协作者，决策者

10.2.2 变更处理过程

接到变更之后，就将对其进行一系列的处理，主要是对业务、技术、项目的影响程度进行分析，从而决定是否接受变更，确定该变更的优先级，以及响应的策略与具体安排。



1、业务影响度分析

（1）业务影响度分析

首先确定变更响应的必要性，然后

一、确定影响范围；行为需求类变更；数据、非功能、规划类变更

二、选取正确的评价者

三、对变更做出评价

（对目标、Stakeholder 关注点的贡献度、对现有业务的影响程度，也就是对每个需求从合理性、必要性、影响度，优先级等几个方面进行评价）

（2）技术影响分析（对原有架构、原有人工制品的影响）

（3）项目影响度分析（项目时间、进度、成本）

（4）是否打破基线（考虑变更的重要性和与其他开发工作的相关性）

10.3 变更管理要点二：统一平台

10.3.1 变更管理平台的选择

选择标准：

1、需求变更的生命周期管理

2、有效的权限管理

3、变更的分类与分析功能

10.3.2 变更管理平台的应用要点

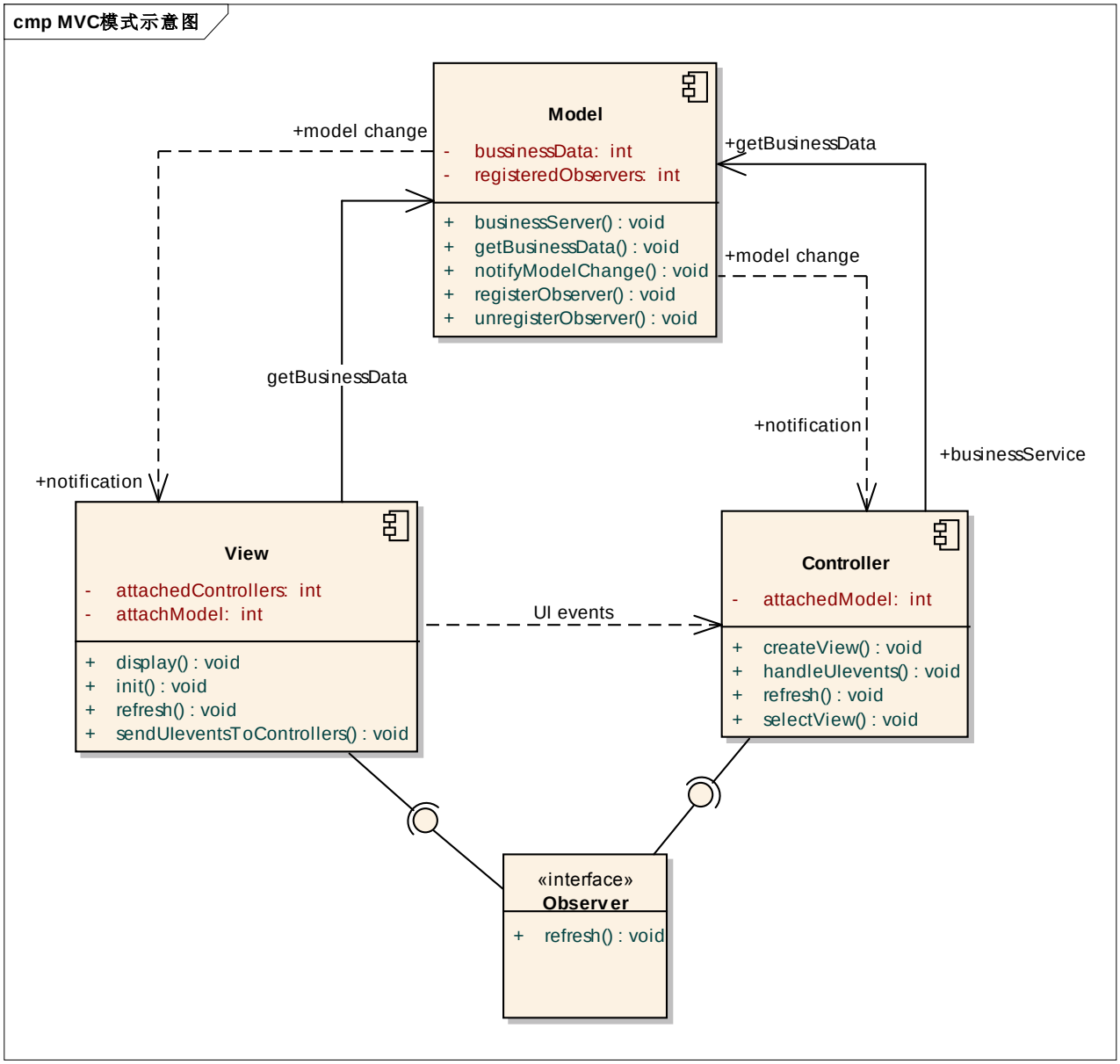
10.4 小结

第 11 章 需求跟踪操作实务

11.1 需求跟踪的基本概念

目标是：

更好的管理需求的状态，更好的分析需求变更产生的影响。


cmp MVC模式示意图
Model
- bussinessData: int
- registeredObservers: int
+ businessServer() : void
+ getBusinessData() : void
+ notifyModelChange() : void
+ registerObserver() : void
+ unregisterObserver() : void
+model change
+getBusinessData
+model change
getBusinessData
+notification
+businessService
+notification
View
- attachedControllers: int
- attachModel: int
+ display() : void
+ init() : void
+ refresh() : void
+ sendUIeventsToControllers() : void
UI events
Controller
- attachedModel: int
+ createView() : void
+ handleUIevents() : void
+ refresh() : void
+ selectView() : void
«interface»
Observer
+ refresh() : void