Chapter 7: 需求建模(三)

Autumn, 2022 @ BJUT

Weidong Wang (王伟东)
wangweidong@bjut.edu.cn

上讲主要内容回顾

(1) 面向对象系统分析的过程、了解类和对象的概念、类与对象的关系等

(2)系统对象与类图的设计

(3)活动图、状态图、顺序图和协作图绘制方法、动态建模的步骤

(4) 清楚在实际的建模中什么时候该使用类图、状态图、活 动图、顺序图和协作图

本讲目录

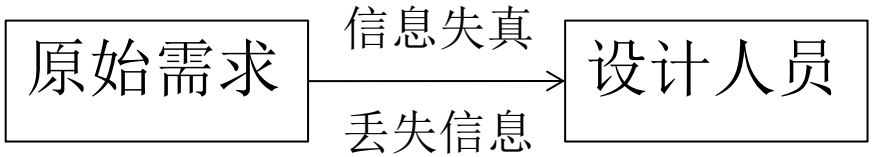
7.1 用例建模

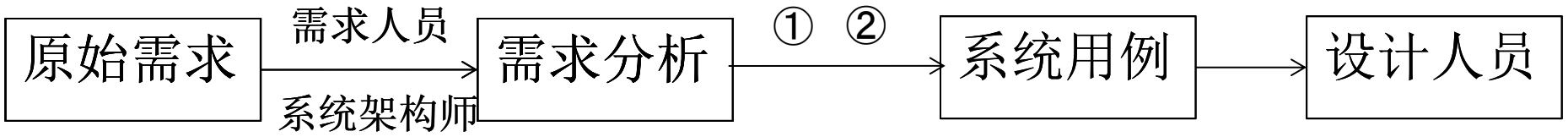
- 业务用例场景
- 获取概念用例
- 获取系统用例

7.2 需求规格说明

7.1 用例建模

概述：





① 业务用例：对现实业务的一种直观理解，没有加入其它因素，容易在客户和开发双方达成共同理解

② 概念用例:用于分解较大的业务用例，从中找到关键和核心的工作单元，针对这些工作单元来简化业务，帮助我们从业务用例过渡到系统用例

③ 系统用例：可以省略系统。就是平时讲的用例，定义为系统既定功能及系统环境模型，作为客户和开发人员之间的契约，贯穿整个系统开发的一条主线

7.1.2 业务用例场景

构建工具（一）：UML 活动图

背景：活动图侧重于描述参与业务的各个参与者在业务当中所执行的活动

优点：适合于分析参与者的职责，并且也有利于将业务用例分解成更小的单元，获取概念或系统用例。

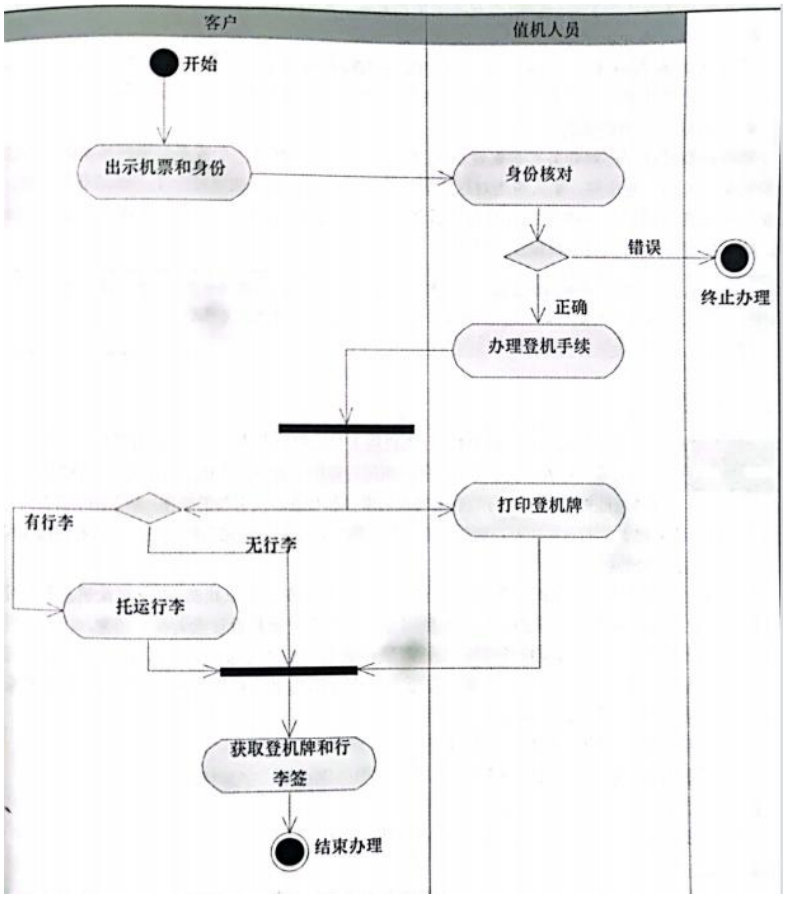
一般情况下，活动图作为描述业务用例场景的必选方式，而将时序图和协作图作为辅助。原因：分析业务阶段，重要的内容就是要得到业务参与者的职责。

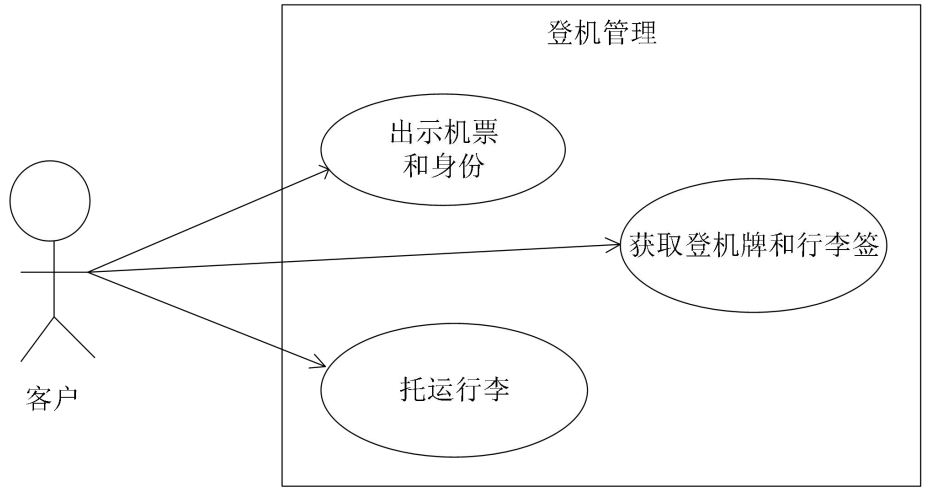
方法：

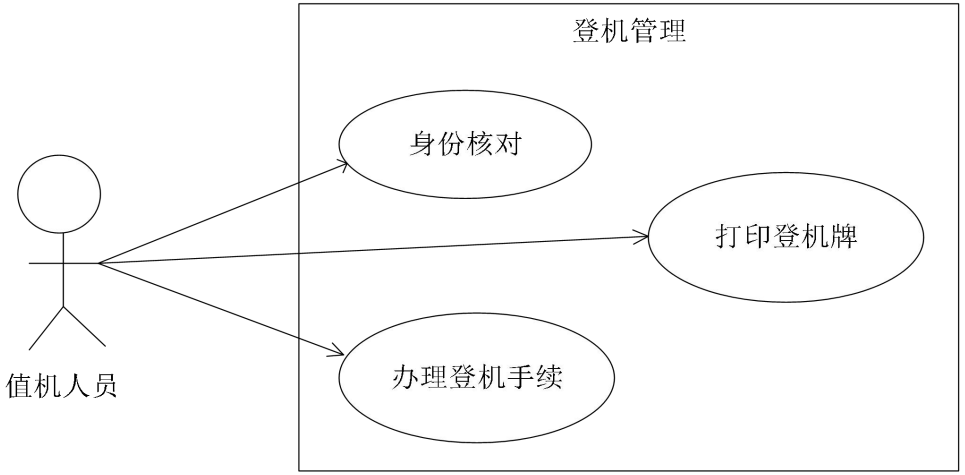
- 参与者和业务工人作为活动图的泳道，将参与者和业务工人所完成的工作作为活动。
- 依据实际业务流程中的执行顺序将这些活动连接起来，形成业务用例场景

注意：一是必须忠实于真实业务；二是一个场景只能描述业务的一种执行方式。

7.1.2 业务用例场景


客户
值机人员
开始
出示机票和身份
身份核对
错误
终止办理
正确
办理登机手续
有行李
无行李
打印登机牌
托运行李
获取登机牌和行李签
结束办理


登机管理
出示机票和身份
获取登机牌和行李签
托运行李
客户


登机管理
身份核对
打印登机牌
办理登机手续
值机人员

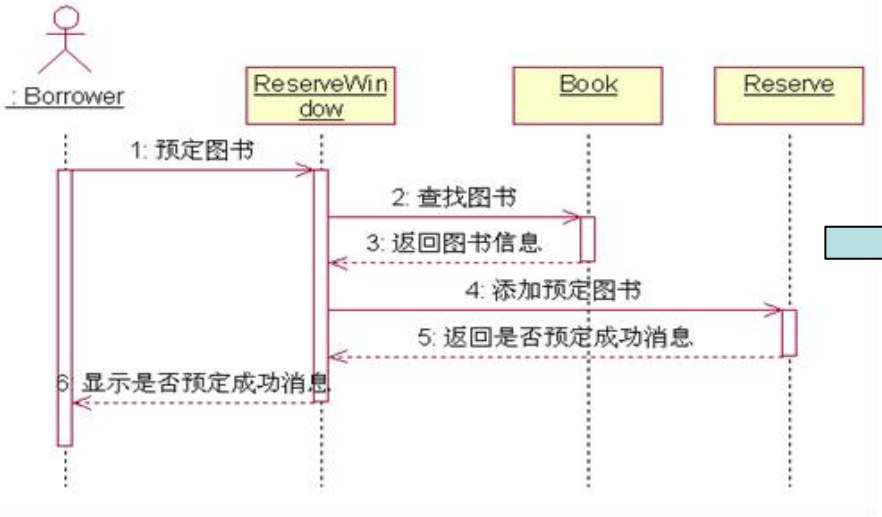
7.1.2 业务用例场景

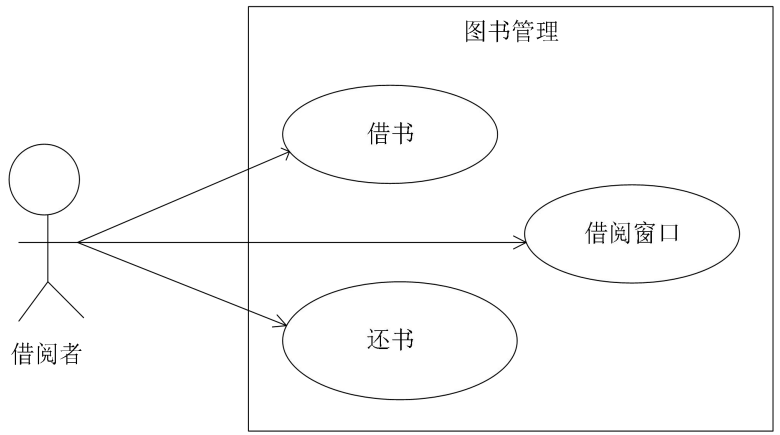
构建工具（二）：UML 时序图

背景：时序图用于描述对象之间按一定顺序互通消息而完成一个特定的目标

与活动图对比：一个强调职责、一个强调顺序之外，还有差别吗？

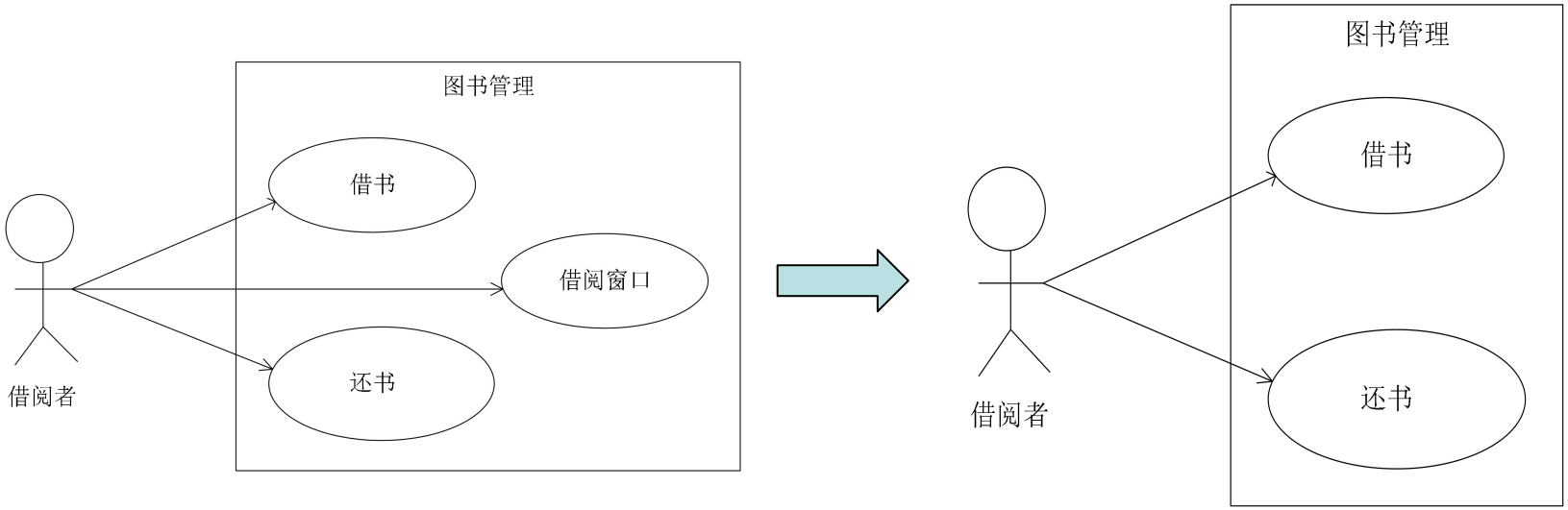
活动图的主要内容是业务主角和业务工人主要做什么；时序图中消息是主要内容，表达的内容是业务主角或业务工人之间传递的是什么。前者表示做了什么完成用例目标；后者表示传递了什么完成用例目标，二者相辅相成。





7.1.3 获取概念用例

概念用例：介于业务用例与系统用例之间，起到桥梁的作用。
比如：业务用例描述的业务很粗略，而系统用例有些时候甚至小到一次计算机交互的粒度。从一个很粗粒度的业务用例过渡到很细粒度的系统用例存在着很多困难。而试图缩小一些业务用例的粒度，则又会导致业务用例数据激增。
需要一种方法分解业务用例，找到关键和核心的工作单元，针对这些工作单元建立模型简化业务，理解业务用例，得到一组缩小了粒度的用例。



7.1.3 获取概念用例

建模示例中有很多业务用例。思考一个问题：
如何从大量的业务用例入手获取概念用例？
最核心的业务是什么？

- 图书馆：借书、还书
- 供电企业管理：管理用电用户、计算电费、收取电费、形成供电收入

综上，撑起图书馆业务的主线，供电企业业务的主线。
另外，如果有很多业务用例都与某个业务主线环节有关，那么我们也需要视情况挑选出一到两个具有代表性的典型的业务用例出来就可以了。

7.1.3 获取概念用例

获取概念用例途径

- 观察现有的业务用例场景，发现那些有着相似名称，在不同的业务用例场景中多次出现或者位于不同泳道中的活动。这些活动很可能就是关键工作单元，以此获得备选概念用例
- 通过对客户业务的分析，或者咨询业务专家，得知客户来说最为重要的一些业务实体
- 通过对客户业务流程的分析，或者咨询业务专家，找到影响整个流程成败的关键业务环节，然后了解这些关键业务环节，获得备选概念用例
- 记住，**概念用例**总是在许多业务场景中**决定场景成败，控制场景进程**，产生和控制最为重要的业务实体

7.1.3 获取概念用例

概念用例适用场合：

- 业务领域规模庞大，业务用例粒度较大，不容易过渡到较小的系统用例
- 业务用例网状交叉，有跨业务的业务用例存在
- 某个业务场景过于复杂，步骤和分支太多，使用活动图绘制用例场景困难
- 有超过7、8个甚至更多的泳道存在
- 你想要在项目早期获得系统原型
- 首次开发这样的系统

概念用例不适用的场合：

- 业务用例规模较小，粒度较小，很容易过渡到系统用例
- 业务用例之间无交叉
- 业务用例场景简单，一般不超过10个步骤
- 非首次开发这样的系统

7.1.4 系统用例

系统用例：系统既定功能及系统环境的模型，作为客户和开发人员之间的契约。

用例是贯穿整个系统开发的一条主线。系统功能性需求完全由用例模型来表达。

用例场景：用例场景说明参与者如何与计算机（即代表了计算机逻辑的分析对象）之间的交互以达成其目的，可以使用任何一种交互图来描述

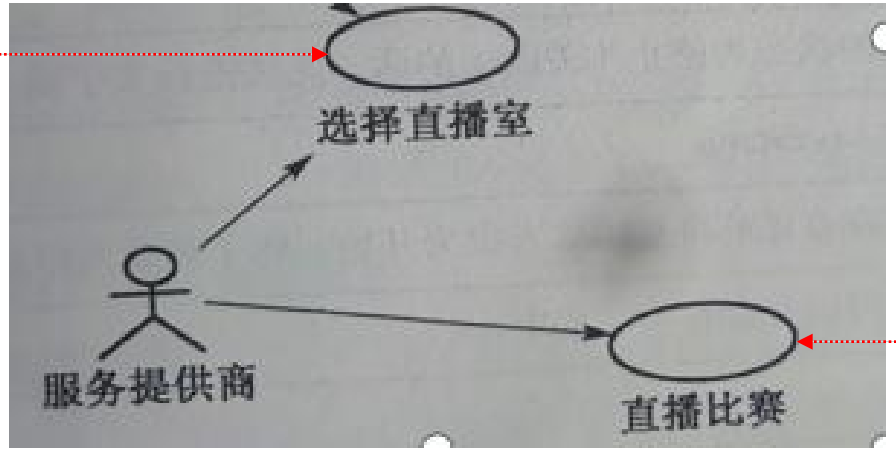
用例视图：用例视图包括参与者与用例，是系统功能性需求的高层视图。功能性需求的全部

用例规约：采用文档形式描述参与者如何启动和终止用例，参与者如何使用用例完成目标，用例的执行事件流，相应规则等内容

补充规约：说明与用例相关的非功能性需求。例响应时间、可靠性、可用性等

业务规则：客户执行业务必须遵守的法律法规、惯例、各种规定，也可能是客户的操作规范、约束机制。可以用伪代码进行编写

7.1.4 系统用例



用例视图

用例编号：U0001	用例名：选择直播室	作者：×××
用例描述：用户或者服务提供商选择一个直播室		
执行者	用户或者服务提供商	
相关用例	无	
前置条件	用户或者服务提供商已登录本系统	
后置条件	无	
基本路径	1．用户或者服务提供商选择一个想观看的直播室 2．登录者为用户时，跳转到观看比赛直播页面 3．登录者为服务提供商时，跳转到比赛播放页面	
备选路径	如果用户还未登录本系统，将会跳转到用户登录页面	
字段列表	用户或者服务提供商的身份保存在 session 的 userType 字段	
业务规则	无	
非功能要求	点亮各播放室的标志	
设计约束	无	
遗留问题	无	

用例规约1

用例编号：U0002	用例名：直播比赛	作者：×××
用例描述：服务提供商比赛内容		
执行者	服务提供商	
相关用例	无	
前置条件	服务提供商已经选择了某个直播室	
后置条件	直播完毕后，直播室的状态必须设为停止（关闭）	
基本路径	1．服务提供商将直播室的状态置为播放中 2．文字直播期间，向系统中输入比赛情况 3．直播完毕后，直播室的状态置为停止（关闭）	
备选路径	如果直播室的状态为停止（关闭）的话，无法进行文字输入	
字段列表	直播状态为 LiveStatus	
业务规则	直播者必须在直播前将直播状态设为开始，终了时设为停止（关闭）	
非功能要求	直播的内容可以保存在本系统中	
设计约束	无	
遗留问题	无	

用例规约2

7.1.5 获得系统用例

- 如果分析工作是从业务用例建模开始的，那么系统用例建模时，应当已经有了业务用例模型。很可能不在需要向客户一一询问决定系统用例，大部分用例可以**从业务用例中推导出来**
- **分析业务用例场景，建议用活动图，**方便观察职责（活动）。**一开始把泳道中的活动都作为一个用例，以泳道作为参与者，把它们绘制出来**。考虑以下问题：

（1）**排除用例**

观察候选用例，如果参与者不用计算机来操作这个用例，则可以排除它。

如果参与者希望计算机来使用它，但**计算机环境不允许**（客户并一定明白计算机能做什么），否则与客户沟通**更换实现方案或者放弃**；

另一方面，如果该用例可以用计算机实现，但是**代价巨大**，项目成本无法承受，则与客户沟通**更换方案或放弃**。

7.1.5 获得系统用例

- **合并用例**：观察剩下的候选用例，分析参与者使用目的，从参与者的关心结果可以得出，虽然候选用例可能有不同的名字，但是如果它们的结果是相同的或相似的，应当考虑合并它们。

举例：虽然**审批A文件、审批B文件**是两个不同的候选用例，但是它们的结果都导致某业务得到批准，那么可以考虑**合并为一个审查文件用例**。合并后，**审批A文件和审查B文件是审查文件用例的泛化**。

- **抽象用例**：观察剩下的候选用例，分析参与者使用它们的方式。使用方式可以从用例场景里归纳出来。如果出现以下情况：结果虽然不同，但是使用过程相同，则应当考虑**抽象出一个描述行为的用例**。

举例：查询A报表和查询B报表是两个不同的目的，有不同的结果。但**它们选择查询条件的过程是一样的**，可以考虑抽象出一个**设置查询条件的用例**，查询A报表和查询B报表都**包含这个用例**。

7.1.5 获得系统用例

- **补充用例**：向用例模型中加入那些与业务实现无关，但对系统运行必须的**非业务需求**。例如：**管理用户账号、备份系统数据**等。如果你的分析工作中包含有概念用例模型，那么充分利用它。（提供参考）

注意：

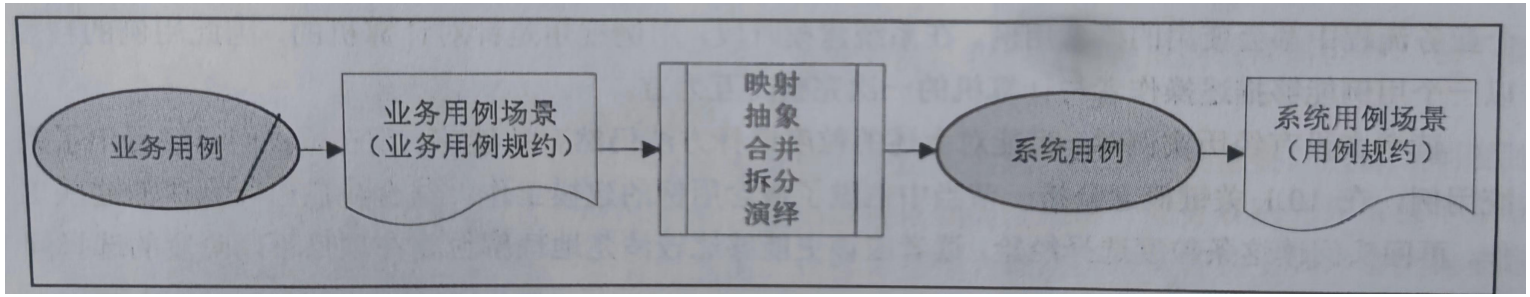
- 系统用例的粒度应当与概念模型用例相当；系统用例的抽象角度应与概念用例相同；概念用例所表述出的核心业务是最需要关心的部分。
- 另一方面，如果用例是从**系统用例开始**的，**客户对计算机环境很熟悉**，那么（需求分析师）不会有太多的困难。否则，**需要花费很大代价去解释他所熟悉的业务在计算机环境里是什么样子，才能获得正确的用例**。

7.1.5 获得系统用例

软件工程中，需求的可追溯性很重要。那么考虑下系统需求如何追溯到业务需求？

- 业务需求是通过业务用例来描述的，以业务用例和业务用例规约为主要文档，描述现实中的业务。系统需求由系统模型描述，以系统用例和系统用例规约为主要文档，描述系统如何映射现实中的业务。对比发现，两者差距大，很可能两者联系不上。因此追溯就需要中间业务的桥梁。

第一个讨论：从业务需求过渡到系统需求



注意：实践中，简单文档记录系统用例的获取过程是非常有意义的，作为项目文档存档，不需要交给客户

7.1.5 获得系统用例

第二个讨论：业务用例和系统用例的粒度

阶段不同，使用不同的粒度

- **业务建模阶段**，用例粒度以每个用例能够说明一件完整的事情为宜，一个用例可以描述一项完整的业务流程
- **概念建模阶段**，用例的粒度以每个用例描述一个完整的事件流为宜，可理解为一个概念用例描述1项完整业务中1个关键步骤
- **系统建模阶段**，用例视角是针对计算机的，因此用例的粒度能够描述操作者与计算机一次完整的交互为宜

举例:某供电企业管理信息系统，业务用例是以整个用电申请业务为粒度的；概念用例是以核心业务当中的关键步骤为粒度的；系统用例基本上是以一次完整的人机交互过程为粒度的。

7.2 需求规格说明

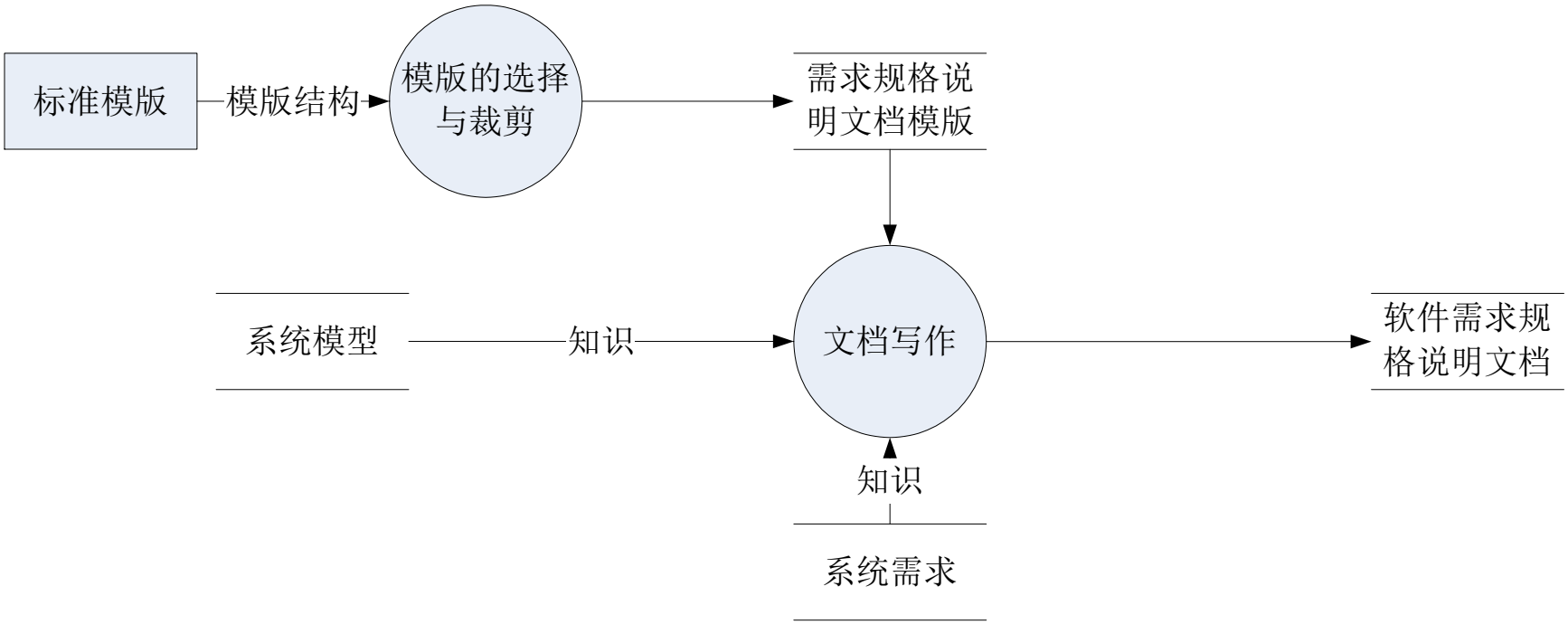
- 需求规格说明概述
- 需求规格说明文档
- 模版的选择与裁剪
- 文档写作技巧
- 优秀需求规格说明文档的特性
- 需求规格说明的实践调查

7.2.1 需求规格说明概述

- 需求获取
  - 目标是得到用户需求——收集需求信息
- 需求分析与建模
  - 目标是更深刻的理解用户需求——界定能够让用户满意的解决方案准则
- 需求规格说明
  - 目标是定义用户需求——准确描述需求及其解决方案

7.2.1 需求规格说明概述

需求类文档写作需要哪些素材？



7.2.2 需求规格说明文档

需求规格说明文档的意义？

① 更好的传递软件系统的需求信息和解决方案给所有的开发者

② 拓展人们的知识记忆能力

③ 作为合同协议的重要部分

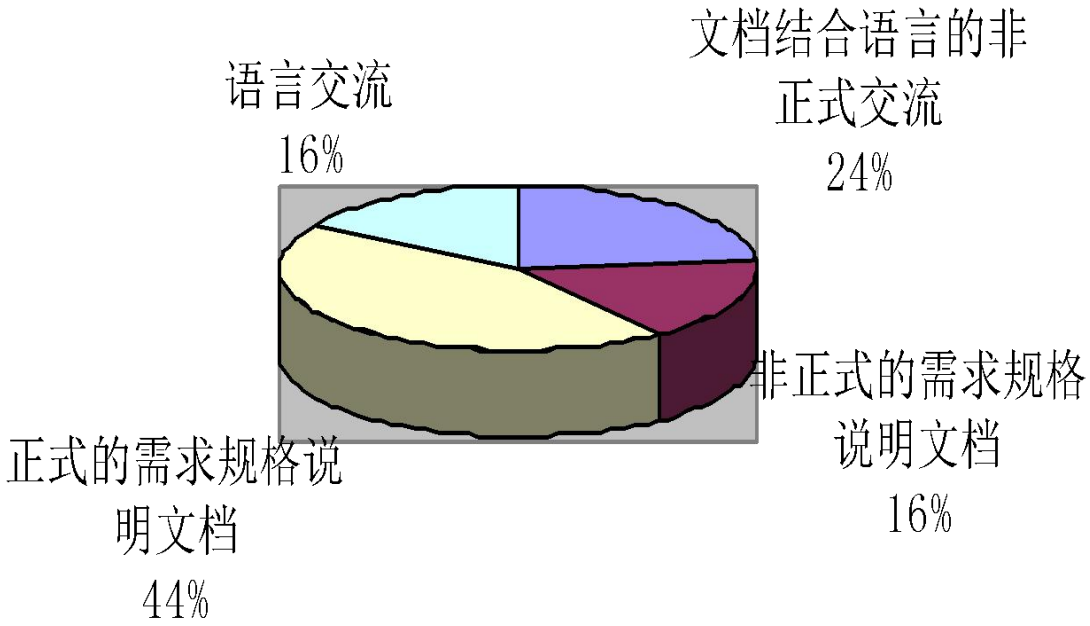
④ 作为项目开发活动的一个重要依据

⑤ 发现和减少可能的需求错误，减少项目的返工，降低项目的工作量

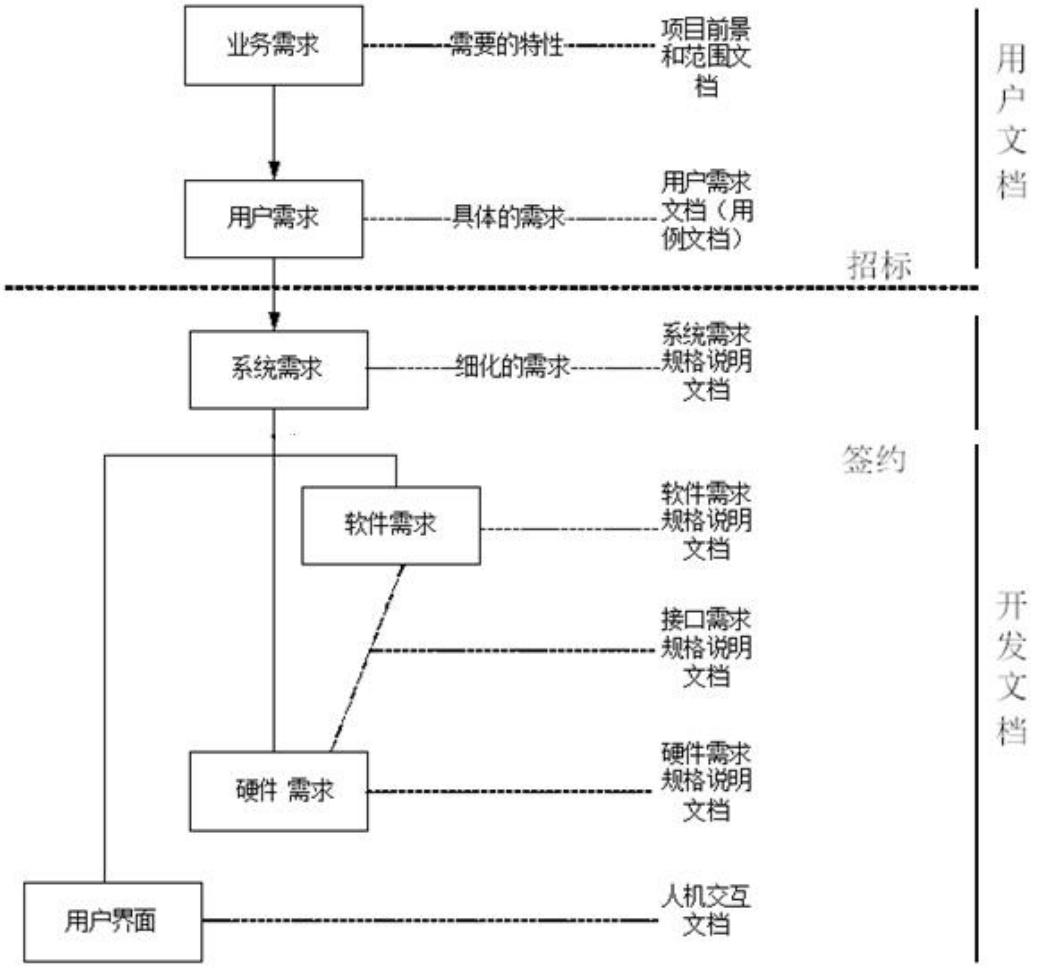
⑥ 作为有效的软件资产

7.2.2 需求规格说明文档

- 交流途径
- 时间压力
- 迭代式开发
  - 敏捷



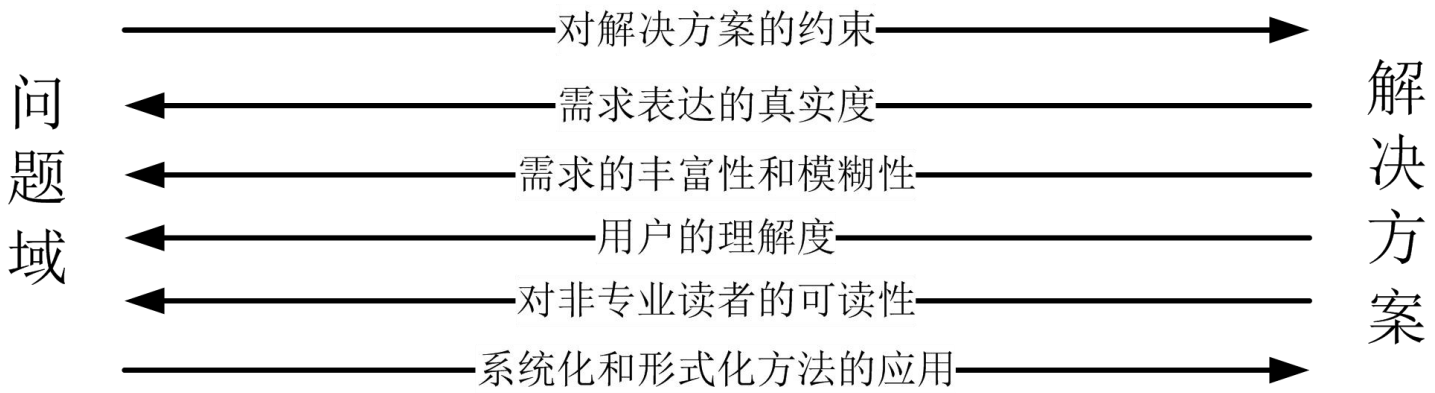
7.2.2 需求规格说明文档



7.2.2 需求规格说明文档

- 类型

项目前景和范围文档	用户需求文档	系统需求规格说明文档	软件需求规格说明文档 硬件需求规格说明文档 接口需求规格说明文档 人机交互文档



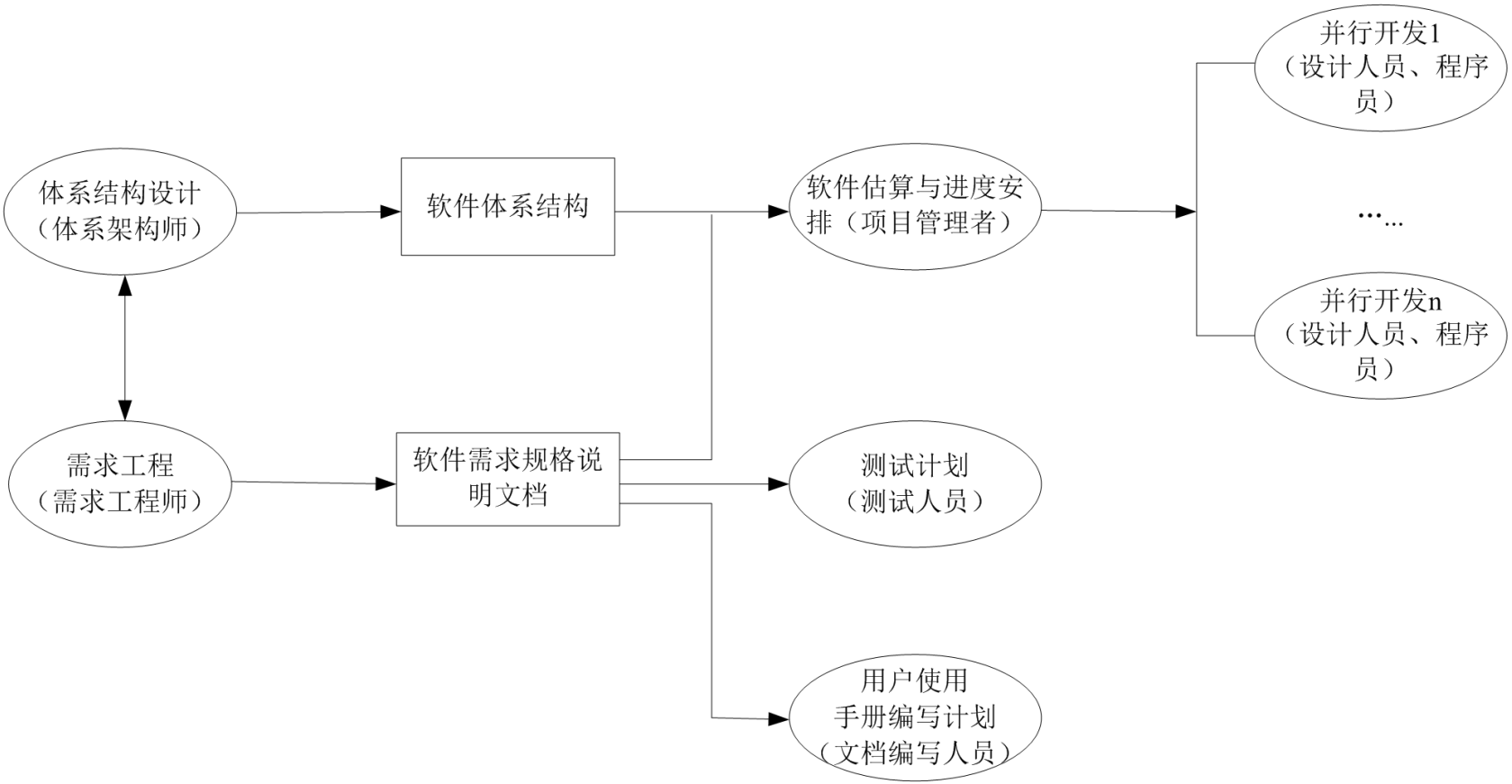
7.2.2 需求规格说明文档

由哪些人员负责或参与完成需求规格说明文档？

- 项目管理者
  - 组织安排、提供条件
- 需求工程师
  - 负责人、主导人
- 文档写作人员
  - 有时会采用，节省需求工程师的时间
- 涉众（用户）
  - 验证人

7.2.2 需求规格说明文档

谁来用？换句话讲，需求规格说明文档谁来使用？



7.2.2 需求规格说明文档

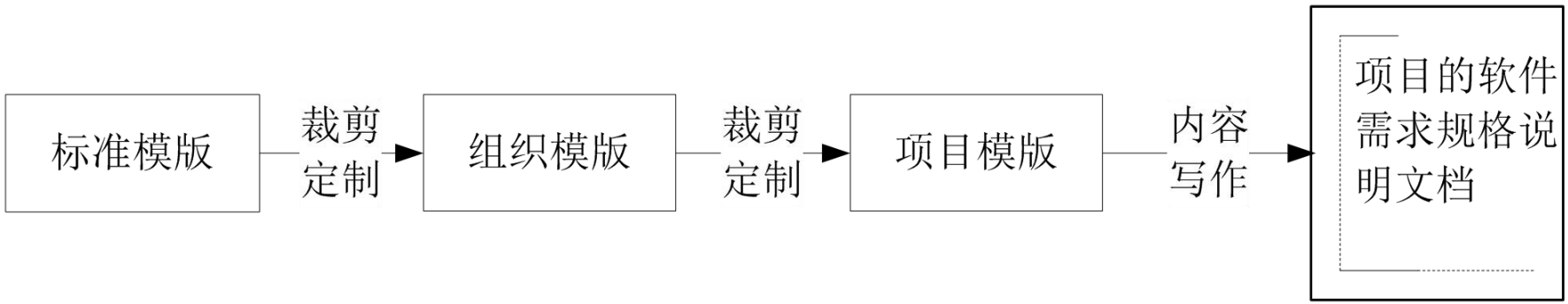
需求规格说明文档采取哪些撰写方法？

- 非形式化
  - 自然语言
  - 限制性文本
- 半形式化
  - 结构化文本
  - 伪码/结构化英语
  - 模型语言
    - 图、表…
- 形式化
  - 形式化语言
    - 数学语言：BNF，Z…

7.2.3 模版的选择与裁剪

- 优秀的文档
  - 结构组织
    - 复用：模版
      - 选择与裁剪
  - 文字写作
    - 字词、句法
      - 写作技巧

7.2.3 模版的选择与裁剪



7.2.3 模版的选择与裁剪 示例

1. 引言
- 1.1 目的
- 1.2 范围
- 1.3 定义、首字母缩写和缩略语
- 1.4 参考文献
- 1.5 文档组织

2. 总体描述
- 2.1 产品前景
- 2.2 产品功能
- 2.3 用户特征
- 2.4 约束
- 2.5 假设和依赖

3. 详细需求描述
- 3.1 对外接口需求
  - 3.1.1 用户界面
  - 3.1.2 硬件接口
  - 3.1.3 软件接口
  - 3.1.4 通信接口
- 3.2 功能需求
  - 3.2.1 系统特性1
    - 3.2.1.1 特性描述
    - 3.2.1.2 刺激/响应序列
    - 3.2.1.3 相关功能需求
      - 3.2.1.3.1 功能需求1.1
      - …
      - 3.2.1.3.*n* 功能需求1.*n*
  - 3.2.2 系统特性2
  - …
  - 3.2.*m* 系统特性*m*
- 3.3 性能需求
- 3.4 约束
- 3.5 质量属性
- 3.6 其他需求

附录

索引

1. 引言
- 1.1 目的
- 1.2 文档约定
- 1.3 读者对象和阅读建议
- 1.4 项目范围
- 1.5 参考文献

2. 总体描述
- 2.1 产品前景
- 2.2 产品特性
- 2.3 用户类及其特征
- 2.4 运行环境
- 2.5 设计和实现上的约束
- 2.6 用户文档

3. 系统特性
- 3.1 系统特性X
  - 3.x.1 描述和优先级
  - 3.x.1 刺激/响应序列
  - 3.x.3 功能需求

4. 对外接口需求
- 4.1 用户界面
- 4.2 硬件接口
- 4.3 软件接口
- 4.4 通信接口

5. 其他非功能需求
- 5.1 性能需求
- 5.2 安全性需求
- 5.3 软件质量属性

6. 其他需求

附录A： 术语表

附录B： 分析模型

附录C： 待确定问题清单

7.2.3 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 模式1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 模式2

…

3.2.*m* 模式*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 功能需求

3.1.1 模式1

3.1.1.1 对外接口需求

3. 1.1.1.1 用户界面

3. 1.1.1.2 硬件接口

3. 1.1.1.3 软件接口

3. 1.1.1.4 通信接口

3. 1.1.2 功能需求

3.1.1.2.1 功能需求1.1

…

3.1.1.2.*n* 功能需求1.*n*

3. 1.1.3 性能需求

3.1.2 模式2

…

3.1.*m* 模式*m*

3.2 约束

3.3 质量属性

3.4 其他需求

7.2.3 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 用户类1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 用户类2

…

3.2.*m* 用户类*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 类/对象

3.2.1类/对象1

3.2.1.1 属性（直接的或继承的）

3.2.1.1.1 属性1

…

3.2.1.1.n 属性n

3.2.1.2 功能（服务、方法，直接的或继承的）

3.2.1.2.1 功能1

…

3.2.1.2.m 功能m

3.2.1.3 消息（收或发）

3.2.2类/对象2

…

3.2.*p*类/对象*p*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

7.2.3 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 刺激因素1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 刺激因素2

…

3.2.*m* 刺激因素*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 信息流

3.2.1.x 数据流图x

3.2.1.x.1 数据实体

3.2.1.x.2 相关处理

3.2.1.x.3 拓扑结构

3.2.2 处理描述

3.2.2.m 处理m

3.2.2.m.1 输入数据实体

3.2.2.m.2 处理的算法或规则

3.2.2.m.3 被影响的数据实体

3.2.3 数据结构描述

3.2.3.p 结构p

3.2.3.p.1 记录类型

3.2.3.p.2 字段构成

3.2.4 数据字典

3.2.4.q 数据元素q

3.2.4.q.1 名称

3.2.4.q.2 表示法

3.2.4.q.3 单位/格式

3.2.4.q.4 精确度/准确度

3.2.4.q.5 取值范围

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

7.2.3 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.n 用户类别n

3.2.n.x 系统特性x

3.2.n.x.1 特性描述

3.2.n.x.2 刺激/响应序列

3.2.n.x.3 相关功能需求

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

7.2.4 文档写作技巧

写作注意事项

- 写作是一门艺术
  - 没有什么固定的规律
  - 有一些效用有限的经验原则
    - 文档的组织方式；
    - 常见情景的处理；
    - 常用的写作技巧；
    - 容易出错的地方等。
- **文档化的目标是交流**
  - 简洁、易读　VS　严格、准确
  - 不要机械的照搬某些标准和规则

需要考虑的问题如下：

有没有另外一种更容易理解的表达方式？
是否一次性提供了太多的信息？
对读者来说什么是重要的，什么是不重要的？
是否太抽象了？需不需要举例说明？
是否太专业了？需不需要解释原理？
会不会引起读者对内容的错误解释？
哪些内容有益于读者？有益于哪些读者？
文档在整体上是不是过于机械、乏味或者松散？
文档枯燥吗？令人厌烦吗？

7.2.4 文档写作技巧

- 所有内容位置得当
  - 借鉴和使用标准的文档模版
- 引用或强化，但不重复
  - 引用而不是复制
  - 强化与重复
  - 引言与冗余元文本

7.2.4 文档写作技巧

- 形式依赖于内容
  - 根据需要表达的内容，选择合适的表达方式
- 使用系统的表达方式
  - 人们倾向于系统的表达方式
    - 使用相同的语句格式来描述所有的细节需求。
    - 使用列表或者表格来组织独立、并列的信息。
    - 使用编号来表达繁杂信息之间的关系，包括顺序关系、嵌套关系和层次关系。

7.2.4 文档写作技巧

- 定义术语表或数据字典
  - 术语不一致
  - “方言”问题
  - 错误术语和冗余术语
- 避免干扰文本
  - “这一段的意思是…”
  - “上一句话是指…”
- 避免歧义词汇
  - 表15－1

歧义词汇	改进方法
可接受的、足够的	具体定义可接受的内容，说明系统怎样判断“可接受”或“足够”
大概可行的、差不多可行的	不要让开发人员来判断“大概”和“差不多”到底是否成立。应将其标记为待确定问题并标明解决日期
至少、最小、不多于、不超过	明确指定能够接受的最大值和最小值
在……之间	明确说明两个端点是否在范围之内
依赖	描述依赖的原因，数据依赖？服务依赖？还是资源依赖？等等
有效的	明确“有效”所意味的具体实际情况
快的、迅速的	明确指定系统在时间或速度上可接受的最小值
灵活的	描述系统为了响应条件变化或需求变化而可能发生的变更方式
改进的、更好的、更快的、优越的	定量说明在一个专门的功能领域内，充分改进的程度和效果
包括、包括但不限于、等等、诸如	应该列举所有的可能性，否则就无法进行设计和测试
最大化、最小化、最优	说明对某些参数所能接受的最大值和最小值
一般情况下、理想情况下	需要增加描述系统在异常和非理想情况下的行为
可选择地	具体说明是系统选择、用户选择还是开发人员选择
合理的、在必要的时候、在适当的地方	明确怎样判断合理、必要和适当
健壮的	显式定义系统如何处理异常和如何响应预料之外的操作
无缝的、透明的、优雅的	将词汇里面所反映的用户期望转化成能够观察到的产品特性
若干	声明具体是多少，或提供某一范围内的最小边界值和最大边界值
不应该	试着以肯定的方式陈述需求，描述系统应该做什么
最新技术水平的	定义其具体含义，即“最新技术水平”意味什么
充分的	说明“充分”具体包括哪些内容
支持、允许	精确地定义系统的功能，这些功能组合起来支持某些能力
用户友好的、简单的、容易的	描述系统特性，用这些特性说明词汇所代表的用户期望的实质

7.2.5 优秀需求规格说明文档的特性

- 完备性
  - 标准
    - 描述了用户的所有有意义的需求，包括功能、性能、约束、质量属性和对外接口。
    - 定义了软件对所有情况的所有实际输入（无论有效输入还是无效输入）的响应。
    - 为文档中的所有插图、图、表和术语、度量单位的定义提供了完整的引用和标记。
  - 前景和范围
  - TBD问题

    To Be Discussed / Determined 的缩写

7.2.5 优秀需求规格说明文档的特性

- 一致性
  - 标准
    - 细节的需求不能同高层次的需求相冲突，例如系统需求不能和业务需求、用户需求互相矛盾
    - 同一层次的不同需求之间也不能互相冲突
  - 评审
  - 自动化检查

7.2.5 优秀需求规格说明文档的特性

- **根据重要性和稳定性分级**
  - 建立需求的优先级
- **可修改**
  - 标准
    - 结构和风格使得人们可以对其中任一需求进行容易地、完整地、一致地修改，同时还不会影响文档现有的结构和风格
  - 文档的可修改性要求：
    - 有着条理分明并且易于使用的组织方式，包括目录、索引和显式的交叉引用。
    - 没有重复冗余。
    - 独立表达每个需求，而不是和其他需求混在一起。

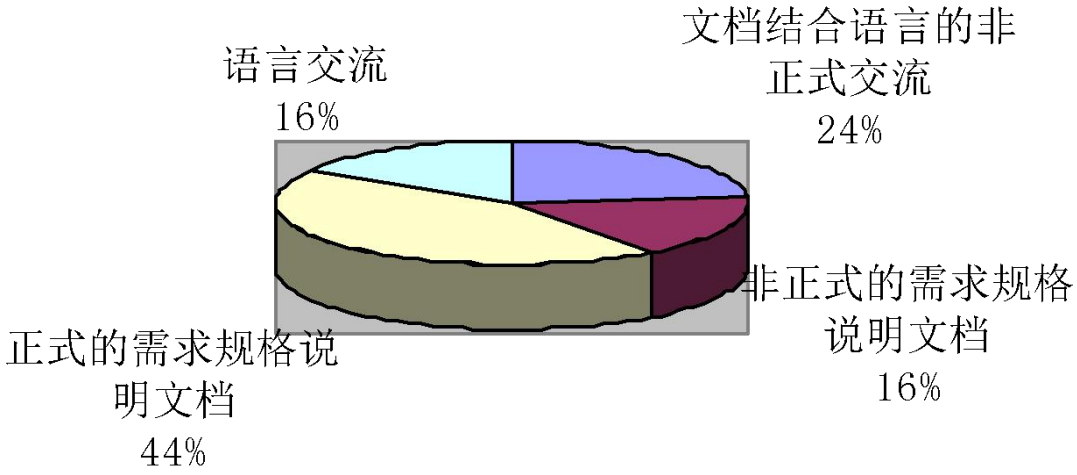
7.2.5 优秀需求规格说明文档的特性

可跟踪

- 前向跟踪（Pre-traceability）
  - 能找到需求的来源，例如和更早期文档的显式关联。
- 后向跟踪（Post-traceability）
  - 能找到需求所对应的设计单元、实现源代码和测试用例等，它要求每个需求都要有唯一的标识或者可供引用的名称

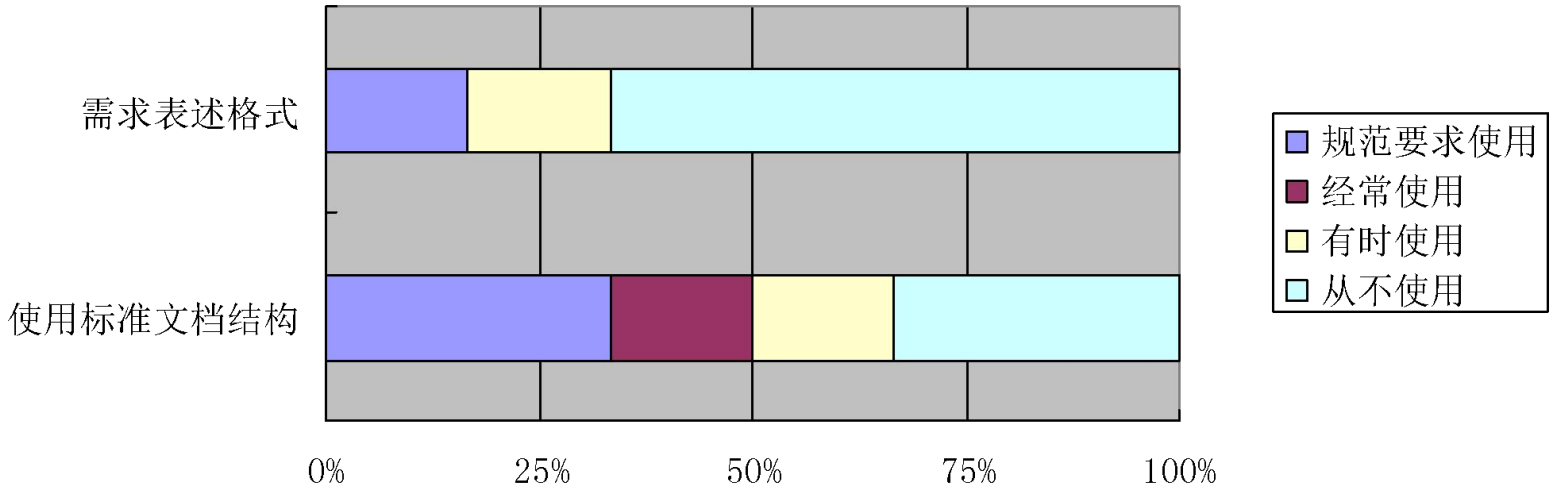
7.2.6 需求规格说明的实践调查

- 需求规格说明文档的编写和使用
  - 时间压力
  - 替代品
  - 迭代式开发



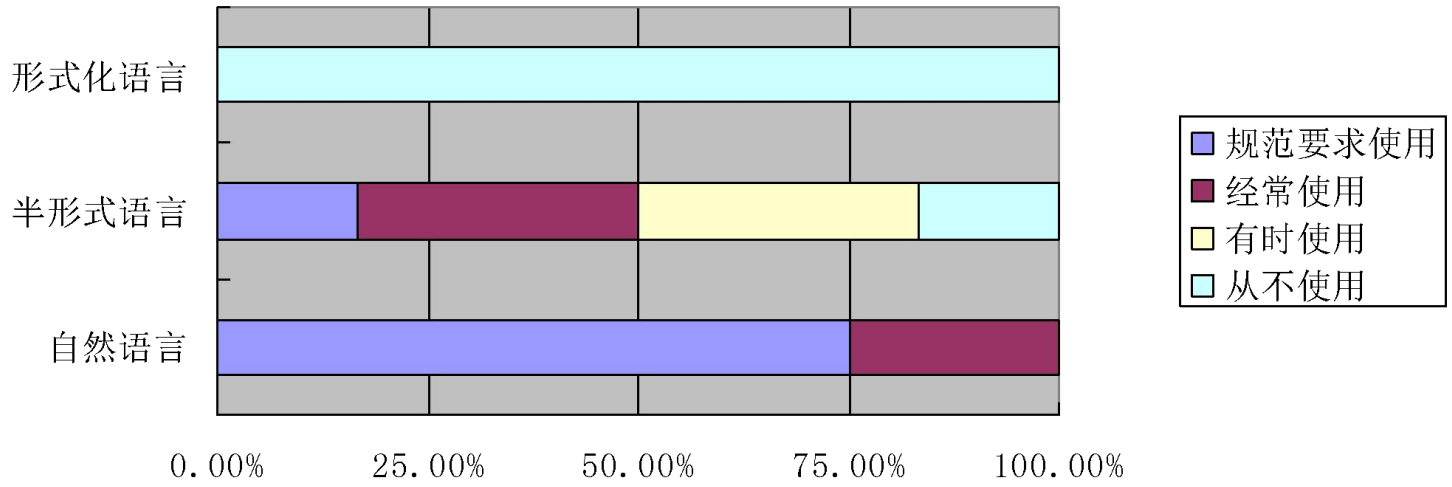
7.2.6 需求规格说明的实践调查

- 模版和示例的使用



7.2.6 需求规格说明的实践调查

- 需求规格说明文档的描述语言



7.2.7 实例分析——共性问题

- 由于时间压力以及采取迭代开发的方式，造成了该项目没有编写需求规格说明书。但是可以采用更为灵活的方式编写，例如wiki、石墨文档。
- 某一预研性质的项目中使用wiki来完成各类文档。结果证明它非常好用。
- 个人认为wiki、石墨文档非常适合用在迭代开发以及预研性质的项目中编写文档。

7.2.7 实例分析——共性问题

- **公司项目的需求规格说明书，主要存在以下几点问题：
  - 模版不是很统一，具有很多个人的特点
  - 没有明确的业务需求、用户需求、系统需求，这三个层次，在需求规格说明书中或多或少地涵盖前三项内容，但显得不够饱满和清晰。
  - 鉴于项目的状况，一般较少考虑硬件需求，一般来讲，项目上线选用的都是最新的硬件设备，成本较高。
  - 内容的书写，自然语言居多，出现歧义、省略、模糊的机会较多，质量不高
  - 从项目的后期来看，性能需求、约束、质量需求没有明确地分门别类地明确列出，导致后期项目中的各个业务流程还是基本可行，但是整体系统还是出现不满足需求的地方。

7.2.7 实例分析—共性问题

- 需求分析报告中夹杂了很多专业名词和行业名词，例如横冲、平衡等等，部分客户看不懂，部分程序员看不懂，只有自己心里明白，但这样就会造成客户和程序员理解上的问题，应该加些注释尽量写得比较白话。
- 另外报告中写得比较凌乱，没有把相关问题归类整合，编写目录，并得到客户的签字确认，导致程序员零散地一条条对着开发，很多地方衔接不是很好
- 另外客户很多想法尤其一些重要部分在软件交付的时候会有所改变，没有签字确认只能自认倒霉。

本讲小结

7.1 用例建模

7.2 需求规格说明

- 需求规格说明定义解决方案和需求
- 掌握文档模版的裁剪技巧和文档的写作技巧
- 优秀的需求规格说明文档需要达到一定的要求
- 实例讲解
- 共性问题


Thank you !