需求部分期末考核（50分）

确定项目前景与范围（10分） – 目标模型：

（业务需求）**问题**：组织的战略目标、利益分配、政策规划、业务流程等高层问题

目标的定义：业务、组织或系统的高层目的，它们说明了系统被开发的原因，并用于指导企业内各种层次上的决策。**（目标是系统被开发的目的。问题的反面，用户的期望）**

目标的描述：类型、属性、链接。

属性：包括**必需属性（名称）、类型、关注、定义（正式与非正式）、优先级、主体、拥有者。**

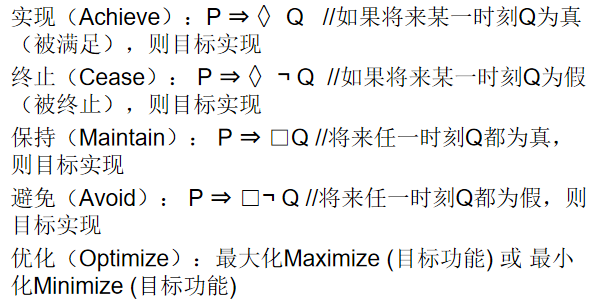
目标类型：

功能目标：描述预期的系统行为，满足型目标（对行为者请求的满足）和信息型目标（为了保持对行为者的信息告知）

非功能性目标：质量目标（）、约束目标（）、安全目标、性能目标、可用性目标

软目标：无法清晰判断是否满足的目标

硬目标：可以通过一些技术确认其是否满足的目标



链接：目标之间的关系和目标与其他模型元素之间的链接

高层次目标：战略性的,全局的, 业务相关的

低层次目标：技术性的，局部的，产品设计相关的

1. 目标之间的关系：
   1. 精化关系：
      1. AND精化关系：如果一系列子目标{G1,G2,…,Gn}的完成有助于目标G的完成
         1. 完备AND精化关系：子目标{G1,G2,…,Gn}的完成能够直接保证G的完成{G1,G2,…,Gn}|=G，那么G与{G1,G2,…,Gn}之间就是完备（Complete）AND 精化关系
      2. OR精化关系：如果任一子目标Gi都是G的替代方案
   2. 阻碍关系：如果子目标O的达成会使得高层目标G失败即O|=¬G，那么O与G的关系就是阻碍关系
      1. AND精化
      2. OR精化
      3. 反向精化
   3. 目标支持与冲突关系：多个目标之间关系
      1. **Support链接**表示一个目标对其他目标的支持作用
         1. 支持关系可以被处理为OR精化关系
      2. **Conflict链接**表示一个目标的实现对其他目标的实现有阻碍作用
      3. ++（Make）：一个目标的成功可以**直接保证**另一个目标的成功；
      4. +（Help）：一个目标的成功可以让另一个目标**更容易成功**
      5. －（Hurt）：一个目标的成功会使得另一个目标的成功**更加困难**；
      6. ― ―（Break）：一个目标的成功会**直接导致**另一个目标的**失败**。
   4. 目标精化的结束条件：子目标展开到**单一事务**时终止

涉众分析（10分） – 涉众识别之ADM模型、涉众评估之Power-Interest模型、涉众共赢之Stakeholder-Issue模型

涉众：所有能够**影响软件系统的实现**，或者会**被**实现后的**软件系统所影响的**，**关键个人和团体**（stakeholder）。

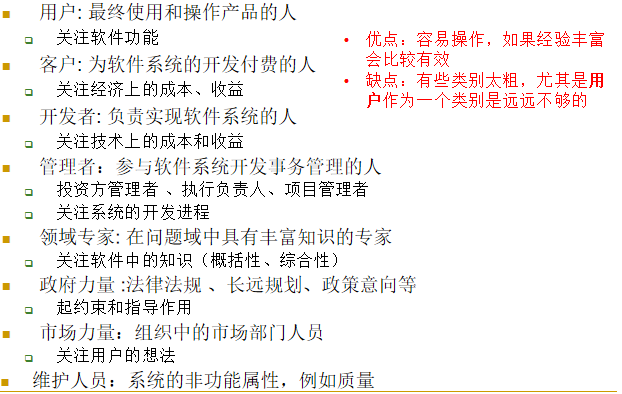
涉众分析围绕一个组织的各个部门内的员工所负责的业务展开。难度随组织机构的复杂性和不确定性增长而增加。

涉众分析过程：

涉众识别：寻找软件系统的涉众分类，**发现比较关键的涉众**

方法：

1. 先膨胀后收缩：尽可能列出涉众后尝试合并，易用但可能有遗漏
2. 检查列表：用户、客户、开发者、管理者、领域专家、政府力量、市场力量、维护人员。



1. 涉众网络：
   1. 寻找最容易识别的初始涉众。
   2. 将初始涉众集中起来，“膨胀 - 收缩”，集体确定关键涉众列表，再请列表中的新代表加入讨论，直到列表稳定

**利用主体依赖模型分析涉众互动，识别关键涉众类别**

主体依赖模型ADM（Actor Dependency Model）

**目标依赖**（goal dependency）：**依赖者希望被依赖者满足一个条件**，但不会规定怎样满足该条件。

**软目标依赖**（soft goal dependency）：一种特殊类型的目标依赖，其条件是无法量化描述的。

**任务依赖**（task dependency）：**依赖者希望被依赖者执行特定任务**。任务依赖比目标依赖更加具体，因为满足目标可以执行很多任务，被依赖者有自己的选择权。而任务依赖直接为被依赖者规定了任务。

**资源依赖**（resource dependency）：**依赖者希望被依赖者提供资源实体（抽象信息或者实物材料）为自己所用**，但不关注提供资源需要被依赖者执行的行为和解决的问题。

涉众描述

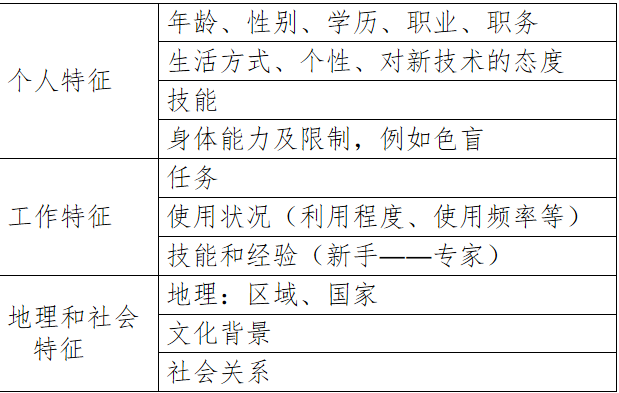
L1：根据软件系统的功能前景寻找涉众

L2：从涉众对象那里获取需求

L3 : 分析涉众的输赢条件,实施共赢策略

L4: 涉众对系统的影响力：了解涉众实现、监控和评估软件系统的能力，分析涉众的力量和影响范围；了解涉众实现、监控和评估软件系统的意愿，即分析涉众的关注点和兴趣取向

L5: 了解涉众的个人特征和工作特征，以便在涉众固定的情况下对软件系统的功能进行合理的调整

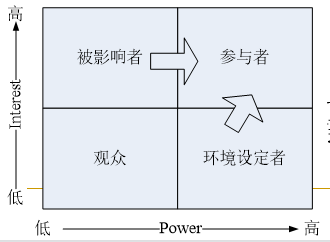


涉众评估

1. 优先级评估：优先考虑涉众的基本特征，尤其是（涉众所完成的）任务特征

基于涉众扩展特征建立的Power/Interest分布图

每个涉众类别都按照自身Power和Interest的高低放在一个合适的位置。



参与者：通常是系统的实际使用者，对系统的最终成功有比较大的影响力。

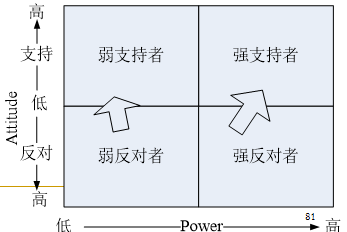
环境设定者：很少直接使用系统，但他们会因为其他因素对系统有较大的影响优先级排第二。

被影响者：有可能是系统的直接使用者，但更有可能是因为系统的出现而被剥夺了部分利益的输家，受系统影响较大，但无法影响系统决策。

观众：不会受系统较大影响，也无能力影响系统的决策。

1. 风险评估：

Power/Attitude分布图



需对强反对者区域的涉众进行仔细分析其高风险因素

共赢分析：为了保证软件系统的最终成功，应尽可能地解决这些冲突，而且最好是在冲突发生之前能够消之于无形。

**步骤：**

1. **发现冲突\*\*Stakeholder/Issue关系图\*\***
   1. **列出系统的所有涉众类别，明确描述他们的兴趣和对系统的期望；**
   2. **从涉众们的兴趣和期望中发现背后涉及的共同问题（Issue）；**
   3. **建立涉众类别和问题的关联，如果某个涉众类别对一个Issue存在兴趣，那么该涉众类别和这个Issue就存在关联关系；**
   4. **对每个Stakeholder/Issue关系，标上该关系所被寄予地期望。**



1. **化解冲突**
   1. **如果某个Stakeholder-Issue关系上所寄予的期望与项目的业务需求无法保持一致，那么它关联的涉众就在该Issue的问题上和项目整体目标存在冲突** 
      1. **涉众和项目负责人互相调整、折中**
      2. **重新评估项目的可行性**
   2. **如果Stakeholder/Issue关系图中某个Issue所关联的不同关系标识有互相冲突的期望，那么就意味着它所关联的涉众在该Issue上存在需求冲突**
      1. **分析各冲突方成为项目赢家的条件**
      2. **适当的调整, 化解冲突**
      3. **分析项目在该Issue上的目标、约束和可选方案，并提供给冲突方进行权衡，促进他们之间协商解决**

涉众代表选择

涉众采样原则：

1. 完整采样: 每种涉众类别都有自己的代表
2. 态度积极: 愿意提供帮助
3. 数量适中
   1. 太少 : 个人看法倾轧群体共同看法
   2. 太多: 达成一致困难
   3. 代表数量的准确数字要视项目的上下文环境来确定, 一般6-10
4. 比例恰当
   1. 计算机技能
   2. 业务技能

用户替代源：因为业务关系而和用户频繁接触的人 ，能够代替他们发表看法

需求获取方法：

1. 面谈

过程：准备面谈、主持面谈、整理面谈报告

优点：1. 简单、成本低；2. 能获得事实、问题、观点等信息；3. 建立友好关系；4. 被会见者会产生参与感和贡献感；

缺点：1. 耗时、时间成本高；2. 会谈者地理分散的情况下难以实现；3. 对参与者的记忆力和工程师的人际交流能力有要求；4. 交谈中产生偏差可能会影响面谈的结果；5. 不了解被会见者认知结构的情况下，难以取得令人满意的效果；

如何进行：

1. 确定背景、主题、会见者；
2. 确定问题；
3. 记录并整理报告；

面谈问题设计：

确认问题类型：

**开放式问题：被会见者对答复的选择可以是开放和不受限制的**

* 优点：

让被会见者感到自在；

会见者可以收集被会见者使用的词汇，这能反应他的教育、价值标准、态度和信念；

提供丰富的细节；

对没采用的进一步的提问有启迪作用；

让被会见者更感兴趣；

容许更多的自发性；

会见者可以在没有太多准备的情况下进行面谈。

* 缺点：

提此类问题可能会产生太多不相干的细节；

面谈可能失控；

开放式的回答会花费大量的时间才能获得有用的信息量；

可能会使会见者看上去没有准备。

**封闭式问题：答案有基本的形式，被会见者的回答是受到限制的**

* 优点：

节省时间；

切中要点；

保持对面谈的控制；

快速探讨大范围问题；

得到贴切的数据

* 缺点：

使得被会见者厌烦；

得不到丰富的细节；

出于上述原因，失去主要思想；

不能建立和面谈者的友好关系。

程序性提示：针对思维特点而设计的面谈问题

1. 总结和反馈
2. 重复和改述
3. 建立场景和细节描述
4. 抗辩

探究式问题：深究答复

诱导式问题：引导被会见者按会见者所想的来回答。

双筒问题：一个问题实际包括两个问题内容

元问题：关于面谈本身的问题

1. 原型

概念：在软件开发中被广泛使用的一种工具，只要是一个展示的一个东西就可以是原型。

使用原型的原因：软件工程中的不确定性

过程：确定原型需求、原型开发、原型评估、原型修正



抛弃式原型与演化式原型：

探索式：以缺陷需求开始继而不断调整和修正需求的原型开发方式称为探索式，要尽可能的调整各种设计选项

实验式：以清晰的用户需求和模糊的实现方法、实现效果、可行性开始，明确需求的可行性和技术实现方案

演化式：以清晰的原型化需求和项目积累下来的原型资产为开始

探索式和实验式产生的原型被称为抛弃式原型。

优点：及早解决系统开发中的不确定性；

风险：成本失控；可能给客户造成错误印象；用户可能被原型的非功能特点遮蔽了双眼，从而忽视了本该重视的功能特性；澄清需求不确定性时可能会掩盖一些用户假设；

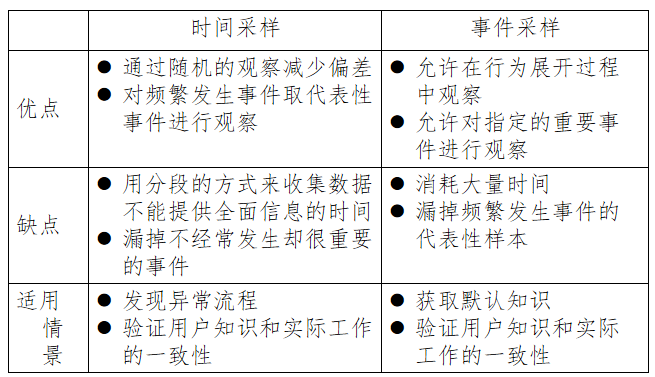
1. 观察和文档审查

观察的适用情况：用户无法完成主动的信息告知的情况下

事件的情境性：某些事件只有和它们发生时的具体环境联系起来，才能得到理解。

情境性使得用户无法有效地主动进行事件的解释和告知；观察法将重点放在问题的上下文；

采样观察



民族志

优点

能够得到信息的深度理解

能够让真实世界的社会性因素可见化

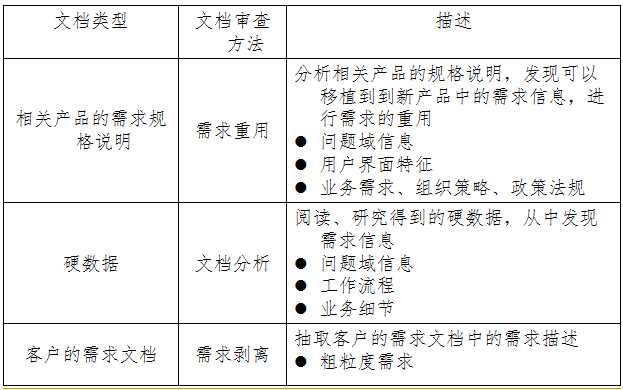
打破人们已有的一些错误假设和错误观念

缺点

需要耗费很多的时间

调研结果很难传递到开发过程

文档审查



专门针对文档进行的需求活动；

需求重用：一旦需求已经成功确定，并且产品本身也成功，就不需要重新开发；

需求分析

1. 需求分析的根本任务
   1. 建立分析模型
      1. 将复杂的系统分解成为简单的部分以及它们之间的联系，确定本质特征

**模型**：对系统进行思考和推理的一种方式

建模方法：

抽象：一方面要求人们只关注重要的信息，忽略次要的内容；另一方面也要求人们将认知保留在适当的层次，屏蔽更深层次的细节

分解：分而治之（将单个复杂和难以理解的问题**分解成**多个相对更容易的子问题，并**掌握各子问题之间的联系**）

投影：多视点方法

* 1. 建立解决方案
     1. 将一个问题分解成独立的、更简单和易于管理的子问题来帮助寻找解决方案



1. 需求分析技术
2. 需求分析方法
   1. 传统分析
   2. 结构化分析：把现实世界描绘为数据在信息系统中的流动，以及在数据流动过程中数据向信息的转化



* 1. 面向对象分析：系统是对象的集合，这些对象之间相互协作，共同完成系统任务。



1. 前期需求阶段的建模与分析
2. 需求分析的活动



* 1. 需求细化：
     1. 明确用户需求的**隐含因素**
     2. 将从**问题域和业务的角度表述的用户需求**等价的**转化**为从**软件和技术的角度表述的系统需求**
     3. 非功能需求也需要从高层次的表述方式**转化**为一系列更加详细和具体的需求表述
     4. 需求细化也会发现**新的细节需求**
     5. 需求已经得了充分的理解，并且开发者已经可以着手为其进行方案设计时停止细化过程
     6. 细化后的需求应该被一一的标识和记录下来
  2. 确定需求优先级
     1. 累计投票
     2. 区域划分
     3. Top-N

N的取值是不受明确限制的，真正受限制的是Top-N个需求的实现代价总和

* + 1. 数据量化



* 1. 需求协商
     1. 明确冲突的因素，避免情绪上的冲突
     2. 明确冲突的解决空间
     3. 确定最佳解决方案



基于UML软件建模的需求细化 – 概念类图、顺序图、状态图

1. 面向对象分析
   1. 面向对象建模：一种用于辨识系统环境中的对象及这些对象之间关系的技术

OMT (James Rumbaugh)

Booch方法(Grady Booch) UML

OOSE (Ivar Jacobson)

* 1. 对象模型、用例模型、行为模型（顺序图、状态图、活动图）、对象约束语言OCL

1. 对象模型
   1. 对象是指在**一个应用当中**具有**明确角色的独立可确认的实体**
      1. 每个对象都要包含标识（唯一）、状态（对象的特征描述，包括对象的属性和属性的取值）、行为（对象在其状态发生改变或者接收到外界消息时所采取的行动）
      2. 事物可以被抽象成为对象的条件：独立可确认和有明确的角色
      3. 链接：对象之间的物理或业务联系
      4. 类：共享相同属性和行为的对象的集合，它为属于该类的所有对象提供统一的抽象描述（接口）和生成模板（实现）
      5. 类产生：类的分类、数据驱动（相同属性）、职责驱动（事物相似性）
      6. 关联：指出了类之间的某种语义联系（\*——\*）
      7. 聚合：部分与整体之间的关系（—— ）
      8. 组合：特殊的聚合，一旦一个部分属于某个整体，那么该部分就无法同时属于其他整体，也无法独立存在（—— ）
      9. 继承（— ）
   2. 领域模型
      1. 概念类：能够代表现实世界事物的概念
      2. 概念类之间存在指明语义联系的关联
      3. 概念类会显式的描述自己的一些重要属性，概念类的属性通常没有类型的约束

建立领域模型

1. 识别候选对象与类
   1. 概念类分类列表
   2. **名词分析**
   3. 行为分析
   4. **CRC**
2. 确定概念类

如果候选对象**既维持一定的状态**，又**依据状态表现一定的行为**，那么它就应该是一个**独立存在**的对象

如果候选对象**只有状态没有行为**，那么就要分析**它的状态是否是系统需要的数据**。（如果**系统需要**它的状态数据，那么该**候选对象就应该作为其他对象的属性**出现在最终的领域模型当中。）

如果候选对象只有行为没有状态，那么往往意味着需求信息的遗漏

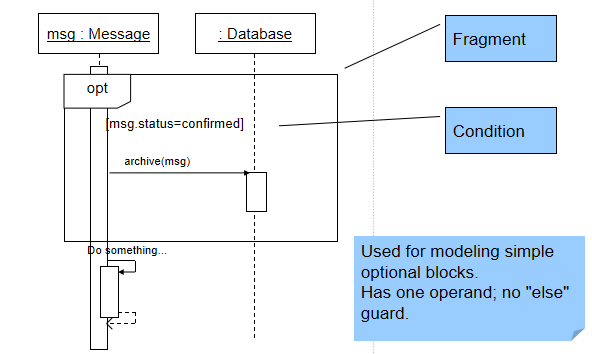
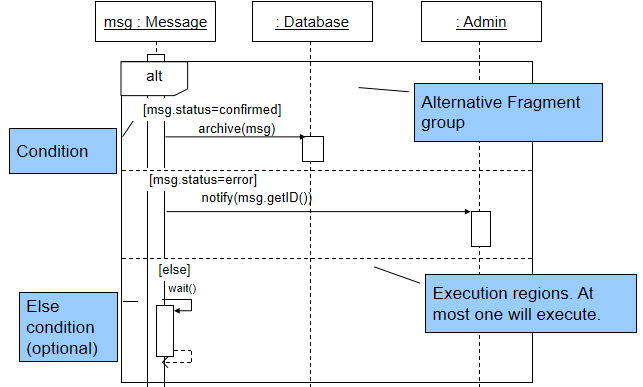
1. 建立类之间的关联
2. 添加类的重要属性
3. （行为模型）用例实现（交互）：顺序图

交互图

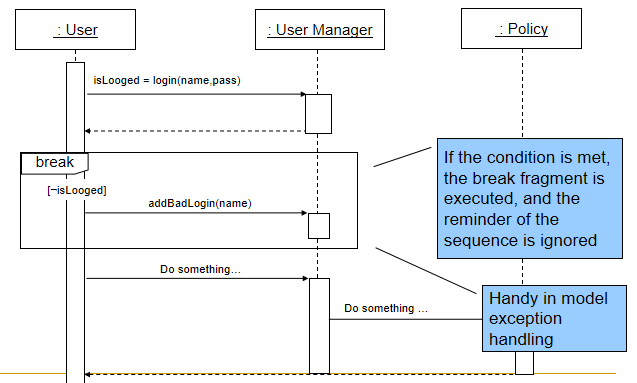
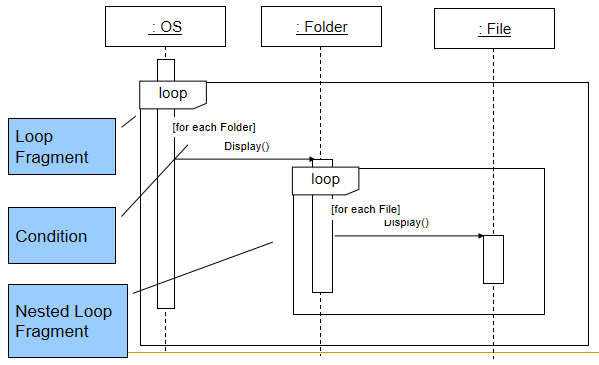
以一组对象为中心的交互描述技术，描述在特定上下文环境中一组对象的交互行为 ，通常描述的是单个用例的典型场景，交互图中的每一个交互都描述了环境中的对象为了实现某个目标而执行的一系列消息交换

交互图片段

options： alternatives:

Loops: breaks:



过程

（1）确定系统顺序图的上下文环境

（2）找出参与交互的对象。

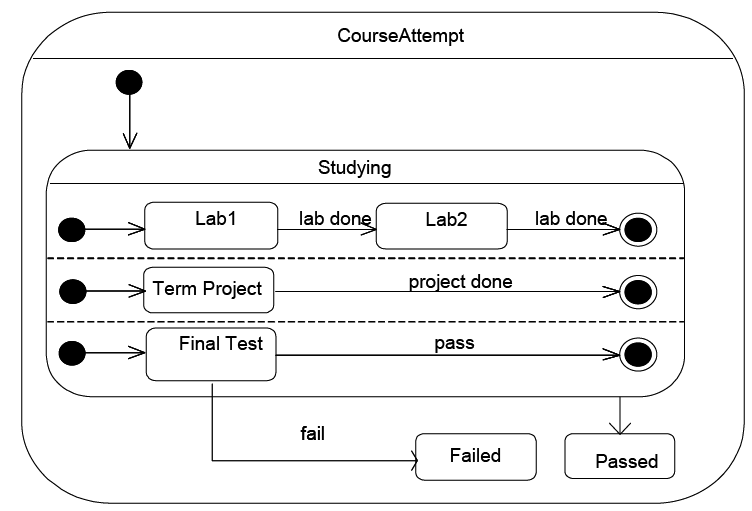
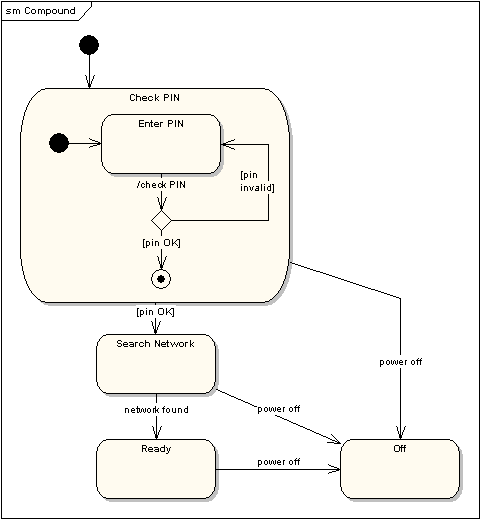
（3）根据发现的对象建立交互图框架。将对象平行排列，并添加对象的生命线。

（4）添加消息，描述交互行为。

1. （行为模型）行为建模：状态图

以状态机理论为基础建立的对系统行为的描述手段

组合状态： 并发状态：



过程:

1. 确定上下文环境 (搞清楚状态的主体常见的状态主体有：类、用例、多个用例和整个系统 )
2. 识别状态，标记初始状态和结束状态(可能会不存在确定的初始状态和结束状态 )
3. 建立状态转换
4. 补充详细信息，完善状态图



1. 对象约束语言OCL
2. CRC

需求验证：专指在需求规格说明完成之后，对需求规格说明文档进行的验证活动。

1. 确保正确地得到需求（需求验证）

2. 确保得到正确的需求（需求确认）

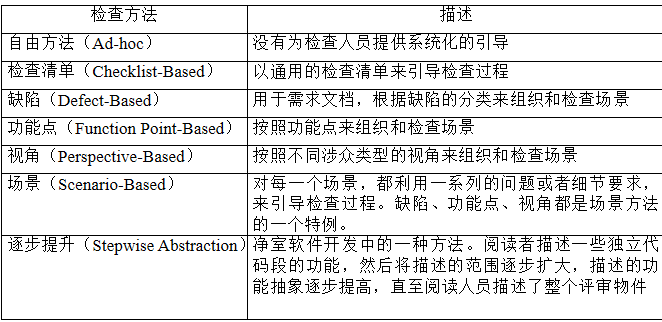
需求验证方法

评审：由作者以外的其他人来检查产品问题。（静态分析手段）

过程



评审检查方法



评审分类



涉及复杂的动态行为时使用原型或模拟方法来验证

开发测试用例：

可能有无法定义测试用例的

1. 排斥性需求

2. 全局非功能需求

用户手册编写

1. 对软件系统功能和实现的描述（功能需求验证

2. 系统没有实现的功能部分（项目范围验证

3. 问题和故障的解决（异常流程需求

4. 系统的安装和启动（环境与约束需求验证

利用跟踪关系

业务需求—>用户需求—>系统级需求

若业务需求和用户需求没有得到后项需求的充分支持，则软件需求规格说明文档存在不完备缺陷。

系统级需求—>用户需求—>业务需求

若不能依据跟踪关系找到一条系统级需求的前项用户需求和前项业务需求，则该需求属于非必要需求

问题的修正

1. 需求澄清

2. 发现缺失需求

3. 解决需求冲突

4. 修正不切实际的期望

需求管理

在需求开发之后的产品生命周期当中保证需求作用的有效发挥。（需求的影响力应贯穿整个后续的产品生命周期）

作用：

1. 增强了项目涉众对复杂产品特征在细节和相互依赖关系上的理解。增强了项目涉众对需求（尤其是复杂需求）的掌握。
2. 增进了项目涉众之间的交流。减少了可能的误解和交流偏差。
3. 减少了工作量的浪费，提高了生产力。需求管理能够更加有效的处理需求的变更
4. 准确反映项目的状态，帮助进行更好的项目决策。需求跟踪信息能够更加准确的反映项目的进展情况
5. 改变项目文化，使得需求的作用得到重视和有效发挥。使得项目涉众认识到需求在项目工作中的重要性

任务与活动：

1. 维护需求基线

交流涉众需要什么。

驱动设计和实现工作。

测试和验证最终产品。

辅助项目管理。

1. 实现需求跟踪

将需求应用、实施到解决方案。

将需求分配到子系统。

1. 控制变更

控制变更。

控制迭代式开发中的变化。

控制变更

需求的变化是正当的和不可避免的

1. 问题发生了改变。

2. 环境发生了改变。

3. 需求基线存在缺陷。

4. 用户变动。

5. 用户对软件的认识改变。

6. 相关产品的出现。

需求变更控制：以可控、一致的方式进行需求基线中的需求的变更处理，包括对变化的评估、协调、批准或拒绝、实现和验证。



变更控制委员会：评价需求的变更，做出批准或者拒绝变化的决定，并确保已批准变化的实现

变更控制中的注意事项：

1. 认识到变更的必要性，并为之制定计划
2. 维护需求基线，审计变更记录
3. 管理范围蔓延
4. 灵活应对变更请求
5. 使用辅助工具