



# 课程章节安排

- 第一部分 软件工程概论
- 第二部分 软件项目开发过程与管理
- 第三部分 软件需求工程
- 第四部分 软件设计
- 第五部分 软件编码、测试与质量保障
- 第六部分 软件实施、维护与演化
- 第七部分 案例分析



## 第三部分 软件需求工程

- 3.1 软件需求与需求工程
- 3.2 需求获取与建模
- 3.3 需求规格说明



## 3.1 软件需求与需求工程

- 软件需求
  - 需求定义
  - 需求分类
- 需求工程



# 软件开发的目标是什么？

- 质量要求：可以工作的软件；
  - 进度要求：在预定的时间完成；
  - 功能要求：完成用户提出的需求；
  - 资金要求：在预算约束下完成；
  - ....
- 
- 软件开发要能够满足各方面的需求



# [案例分析] 学生成绩管理系统需求描述

- 《软件设计与开发实践II》某组同学选择了《学生成绩管理系统》题目，并进行了初步的分析。



# 对于该题目的理解

- 第一，首先在上个学期学完数据库课程后，我们对于数据的管理有了更深刻的体会。其次，上学期的软件实践一中我们也对学生成绩管理系统有所研究。所以，我们认为学生成绩管理系统实际上就是一个数据库管理系统。
- 第二，对于这次的题目，我们小组私下里也讨论过很多次。认为其关键在于如何有效的处理数据，比如数据的插入，删除，排序等等。
- 第三，这次实践中，我们会格外注意数据处理的速度以及界面的整洁美观。



# 初步需求分析

- 对于一个项目，其需求必须符合实际要求。在设计开始之前我们曾经采访过多位同学，他们对于学生成绩管理系统主要有以下需求：
  1. 该系统必须能有效地完成数据的插入、删除、排序、存储等操作。
  2. 该系统对于错误的输入必须有严格的检测。
  3. 针对不同的使用者，该系统最好能给予不同的权限，有些数据对某些人来说是不能查看的。
  4. 该系统的界面必须简单、美观、人机交互性强。



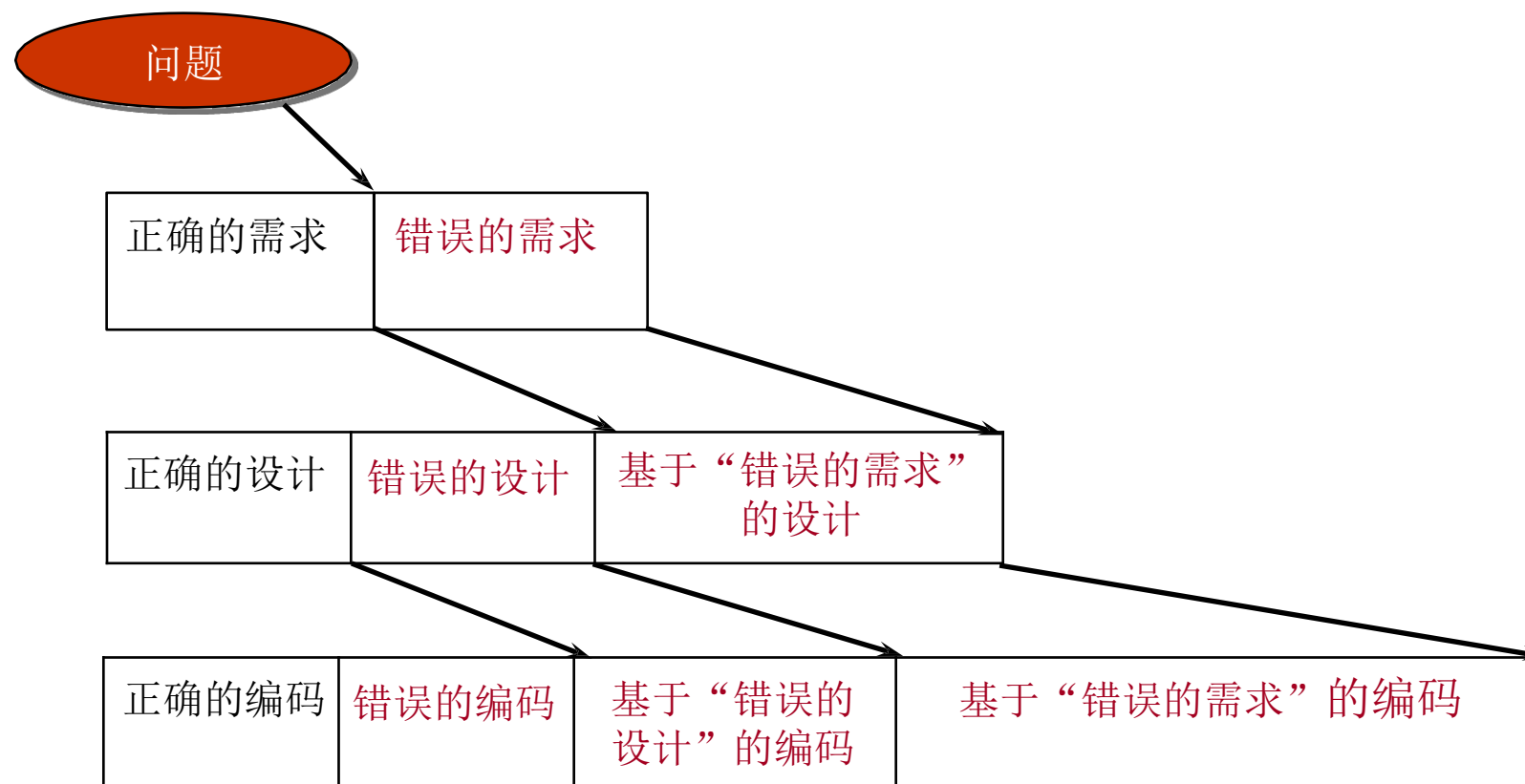
## 关于预期成果

- 首先，该系统完成后应该是一个需密码登陆，根据用户身份的不同，拥有不同权限（界面）人机交互性良好的可视化程序。
- 其次，该系统可以在短时间内完成数据的增删改，排序等操作。
- 最后，也是我们的一点奢望，我们希望能将该系统应用于我们班的成绩管理中，在不断的实践中加以改进。



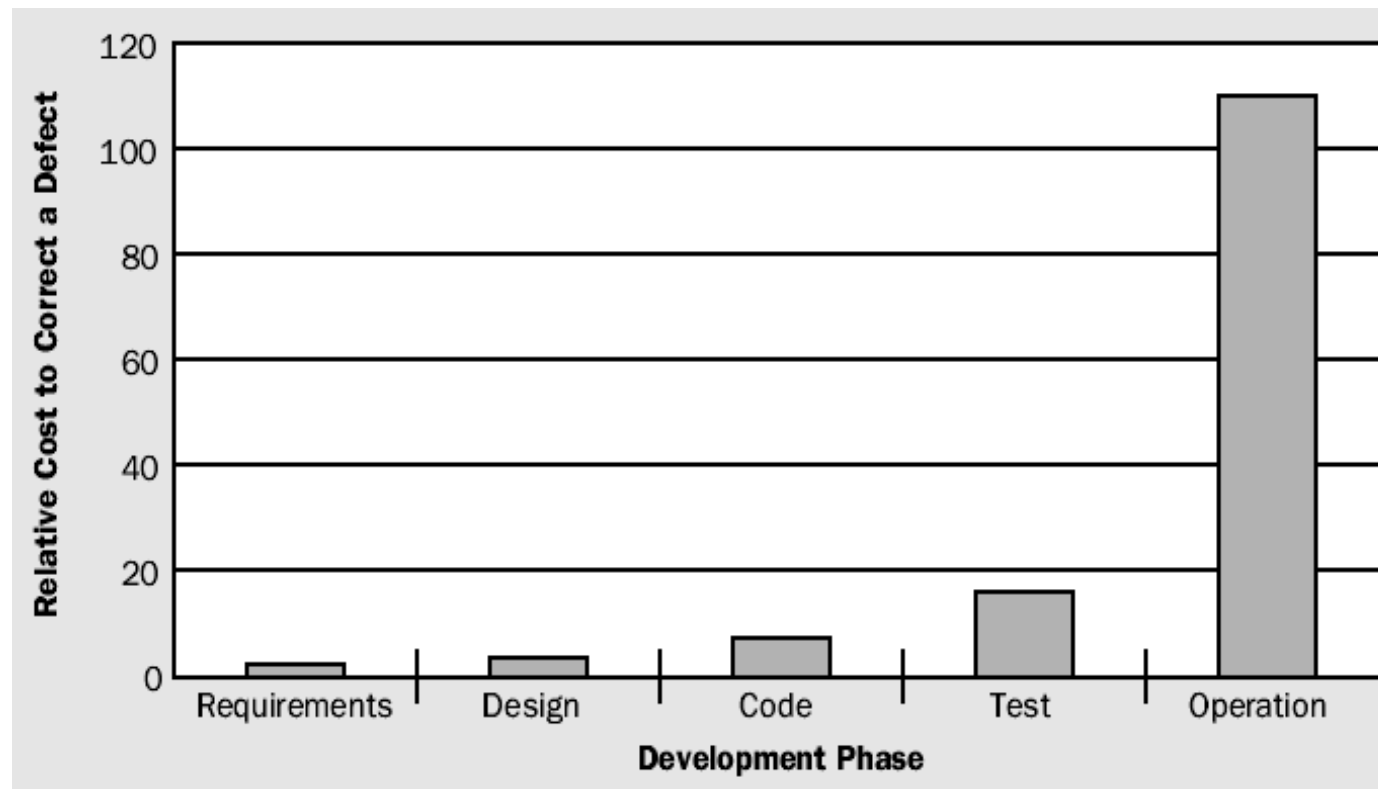


# “错误的需求”的扩散效应





# “错误的需求”的修复代价

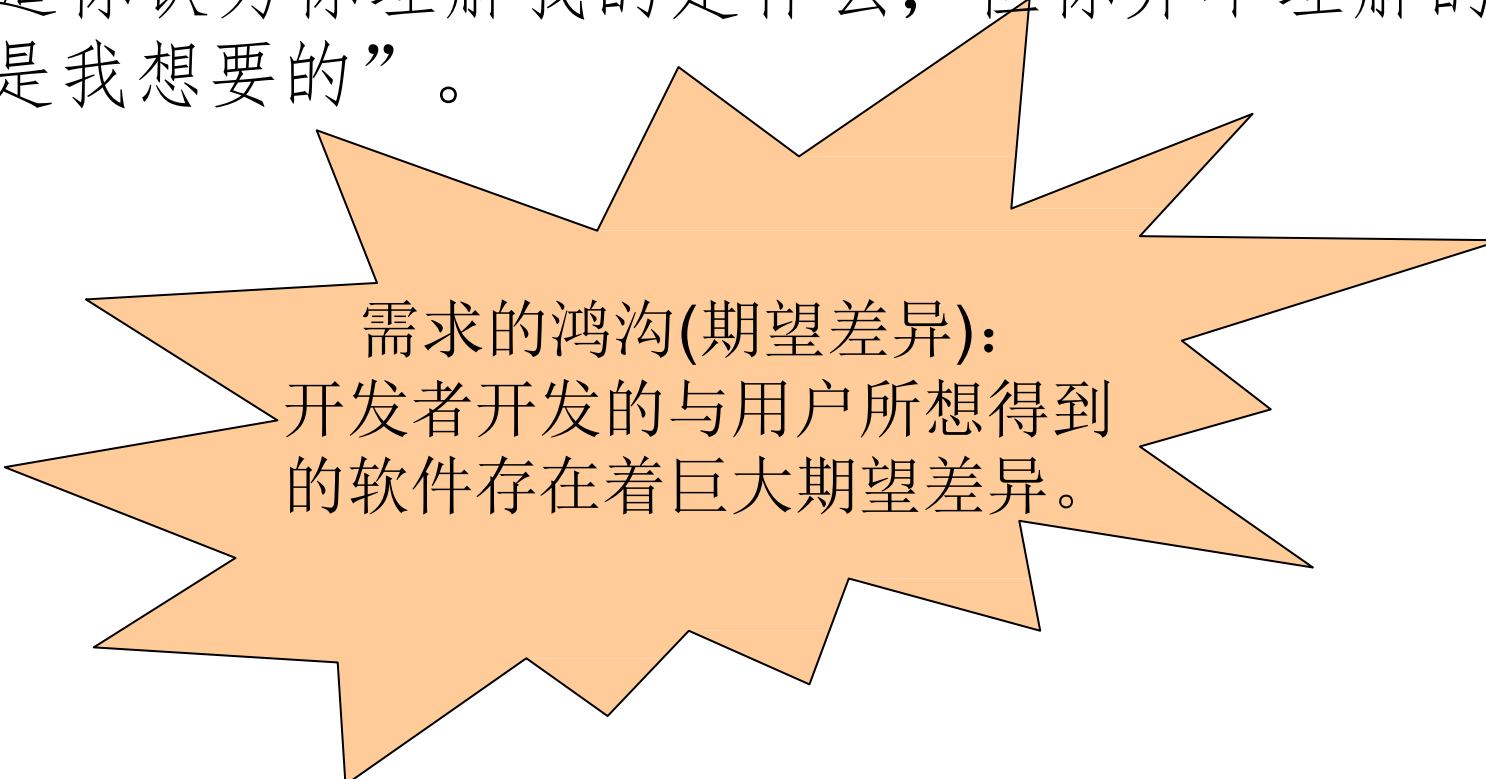


“构建一个软件系统最困难的部分是确定构建什么...在出错之后会严重影响随后实现的系统，并且在以后的修补是如此的困难...”



# 根本原因是什么？

- 恐怖的噩梦：一个客户走进你的办公室，坐下，正视着你，然后说“我知道你认为你理解我的是什么，但你并不理解的是：我所说的并不是我想要的”。



需求的鸿沟(期望差异):  
开发者开发的与用户所想得到的软件存在着巨大期望差异。



# “软件需求”的作用

- “Requirement is the Basics of Quality”
  - 充分理解现实中的业务问题，并作为软件设计的基础；
  - 为软件项目的成本、时间、风险估计提供准确的依据；
  - 减少开发工作量，避免将时间与资源浪费在设计与实现错误的需求上；
  - 通过提供需求文档和需求基线，来有效的管理系统演化与变更；
  - 作为顾客与开发团队之间正式合同的一部分；
  - 为最终的验收测试提供标准和依据；



# 什么是“软件需求”

- **软件需求(Software Requirements):**
  - 用户解决问题以达到特定目标所需的能力;
  - 系统或系统构件要满足的合同、标准、规范或其他正式文档所需具备的能力; ——IEEE, 1997
- **软件需求:** 以一种清晰、简洁、一致且无二义性的方式, 描述用户对目标软件系统在功能、行为、性能、设计约束等方面的期望, 是在开发过程中对系统的约束。
- 需求通常用于表达“**做什么**”, 而不描述“**如何做**”。



# 关于“需求”的例子

## ■ 学生选课系统需求

- 某大学希望采用计算机管理学生的选课
- 学生可以在一个学期开始之前选择该学期开设的某些课程
- 老师可以使用选课系统获得选课学生的名单，并登记学生的课程学习成绩
- 学生不希望自己的学习成绩被他人查阅
- .....(你可以补充吗?)

## ■ 以下描述是否属于需求？为什么？

- 系统通过JDBC与Oracle数据库CourseDB建立连接，并使用T-SQL 语句从CourseOffering数据表中获得课程的开设信息。



## 3.1 软件需求与需求工程

- 软件需求
  - 需求定义
  - 需求分类
- 需求工程

# 需求分类

## 产品/过程

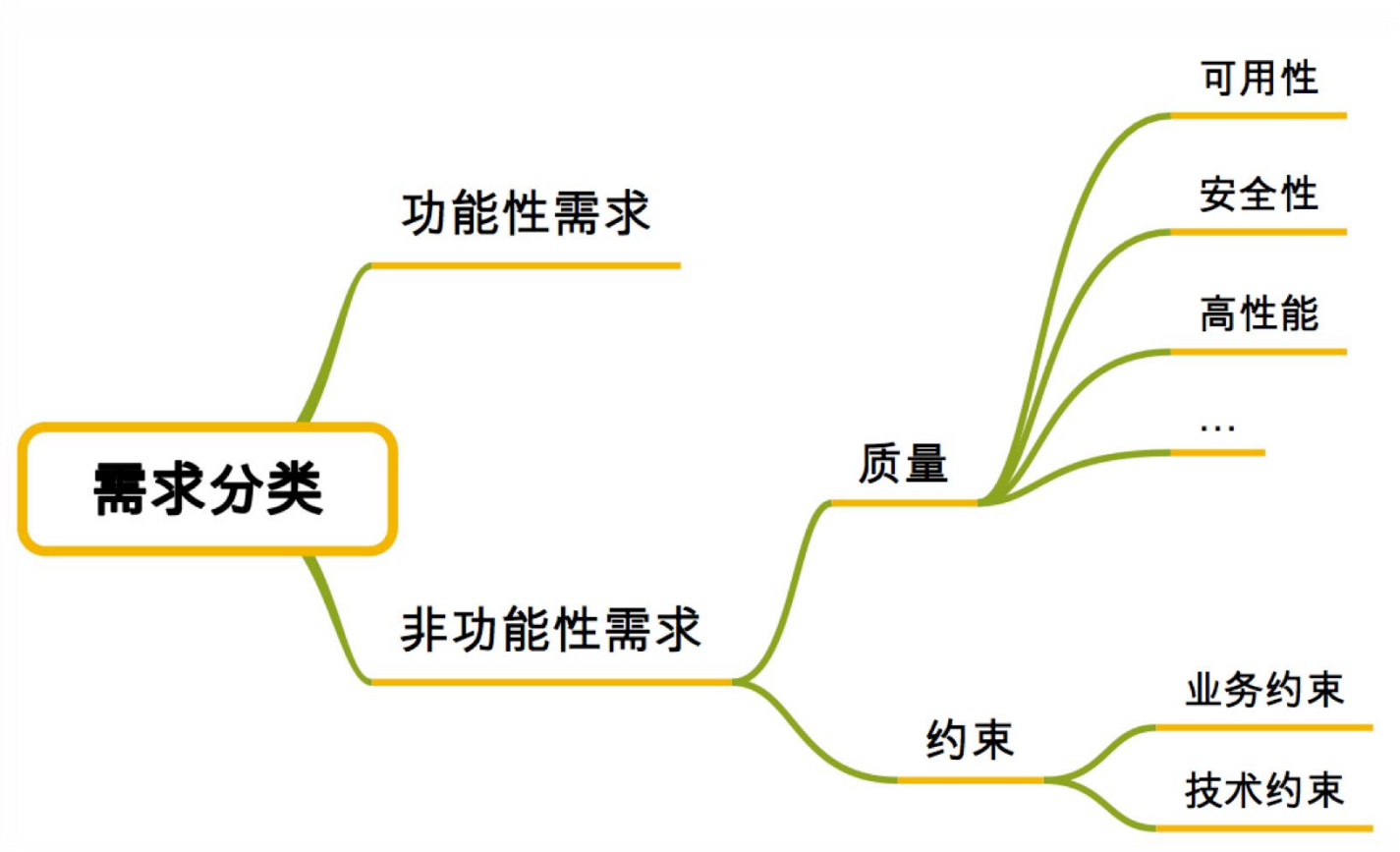
- 产品需求
- 过程需求

## 功能

- 功能性需求
- 非功能性需求

## 抽象层次详细程度

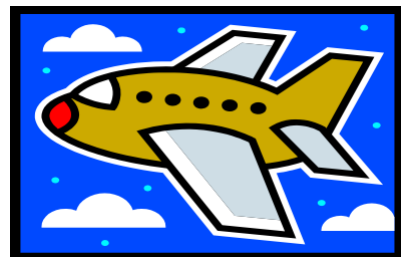
- 业务需求
- 用户需求
- 系统需求
- 软件设计规约





# 功能性需求

- 系统的功能性需求是指满足系统需求需要提供的功能
- 有时，功能需求也被称为“行为需求”



- 举例
- 订票系统需要提供——一个在任何航班上预留座位的功能
- 订票系统需要提供——一个通过信用卡付费的功能



# 1 业务需求

- **业务需求(Business Requirements):** 客户对于系统的高层次目标要求(high-level objectives) , 定义了项目的远景和范畴(vision and scope)
  - 企业的业务需求是关于企业业务的陈述, 和这个需求如何实现无关, 无论是手动的还是通过系统来完成。
  - 业务需求被叫做业务目标。
- **[例] 携程旅行的业务需求是卖飞机票**
  - 公司的目标是成为当人们想买飞机票时首先想到的公司;



## 2 用户需求

- **用户需求 (User Requirements):** 描述的是**用户的目标**，或用户要求系统必须能完成的任务。用例、场景描述和事件——响应表都是表达用户需求的有效途径。
  - 也就是说用户需求**描述了用户能使用系统来做些什么**。
  - 用户需求被称作“用户接口需求”。
- [例]用户可以通过Internet随时查询图书信息和个人借阅情况，并可以快速查找和浏览需要的电子资料；
  - 用户通过Internet查询图书信息；
  - 用户通过Internet浏览个人借阅情况；
  - 用户通过Internet查找和浏览电子资料；



# 业务需求与用户需求的对比

## ■ 针对Course Registration System

### ■ 业务需求

- 由于实行学分制管理，所以希望用计算机管理学生选课。
- 课程信息维护、选课管理、课程成绩登记和查询等业务全部由手工方式改为计算机应用。

### ■ 用户需求

- 教务管理员希望能够增加、修改和删除学校的课程目录，并且设置各学期课程的开设信息。
- 学生希望能够在学期开始之前查询所有开设课程的详细信息，并能够通过校园网进行选课。
- 学生希望在选课期间系统能够24小时使用，系统使用方便快捷。



## 3 系统需求

- 系统需求 (System Requirements, SR): 系统应该提供的 **功能或服务**, 通常涉及 **用户或外部系统与该系统之间的交互**, 不考虑系统内部的实现细节。
  - 企业选择实现系统的首要条件是: 系统需求满足组织的业务需求
- [例]
  - 订票系统需要和用户数据库交互
  - 新的软件会使得汽车的启动速度加倍
  - 我们新产品制造的低成本将会让我们有更高的市场份额并满足销售目标;

# 功能需求的层次性

- 将业务需求转化为系统需求的过程是一个复杂的过程
  - 首先需要分析问题领域及其特性，从中发现问题域和计算机系统的共享知识，建立系统的知识模型；
  - 然后将用户需求部署到系统模型当中，即定义系统行为，让它们联合起来实现用户需求，每一个系统行为即为一个系统需求。
  - 该过程就是需求工程当中最为重要的需求分析活动，又称建模与分析活动。

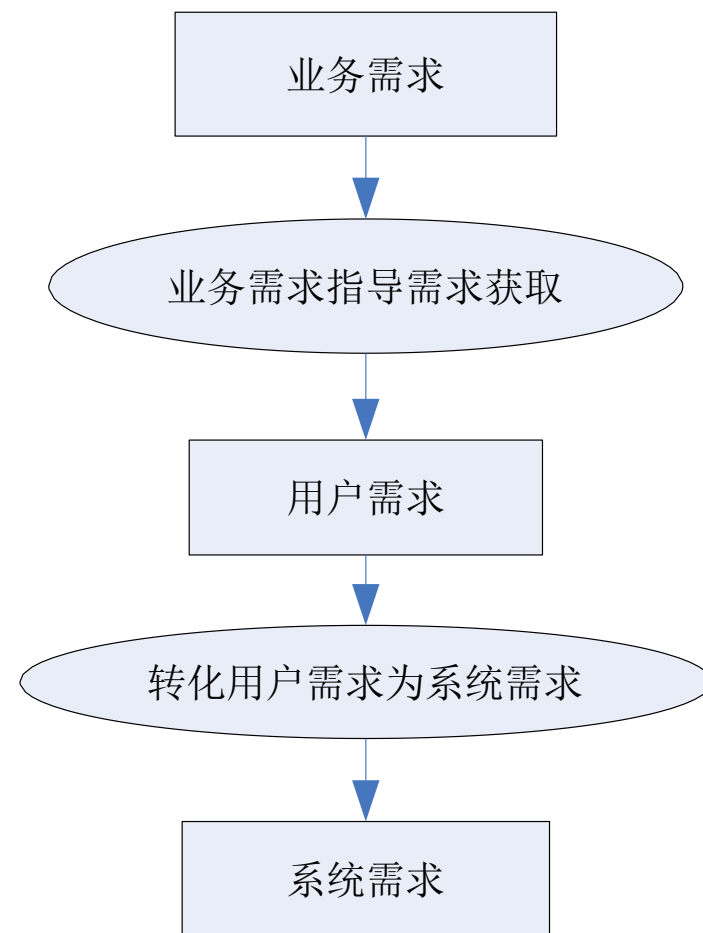


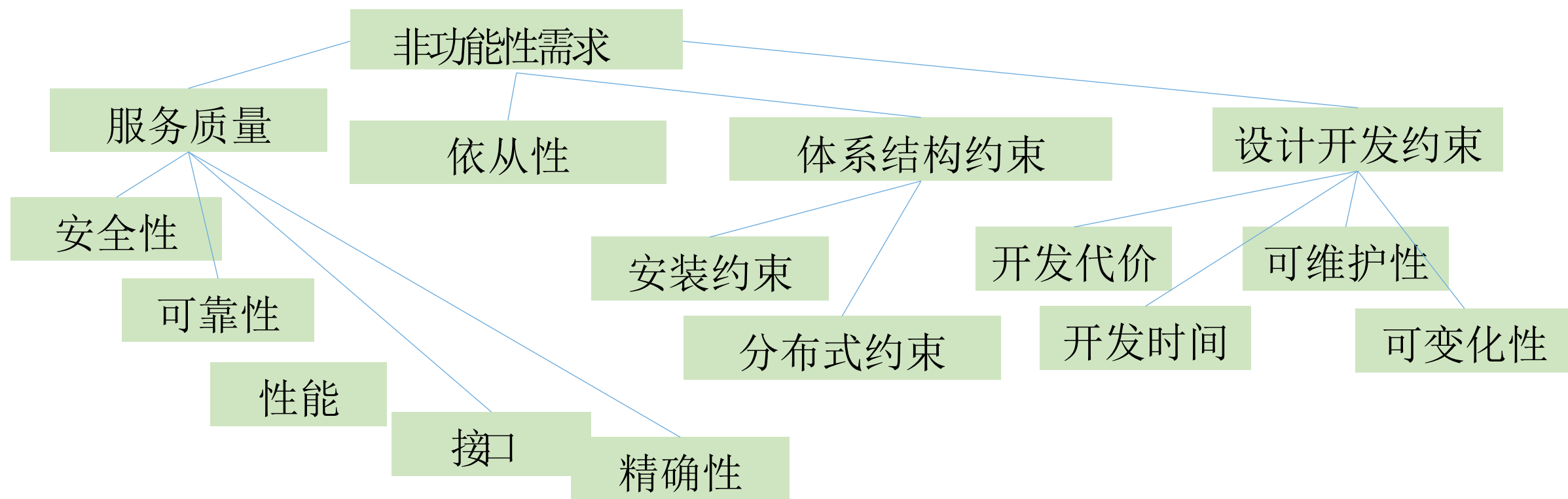
图 不同抽象层次需求之间的联系



# 非功能性需求

## (Non-functional Requirements, NFRs)

- 非功能性需求定义软件系统以及软件开发过程为满足系统功能需求要满足的其他约束条件





# 非功能需求的度量

- 非功能需求：检验起来非常困难，一般采用一些可度量的特性进行描述。

非功能特性	度量指标
速度	每秒处理的事务 用户的响应时间 屏幕的刷新速度
存储空间	分配内存空间、外存空间
可用性	培训时间 帮助页面数
可靠性	平均失败时间 系统无效的概率 失败发生率
容错性	失败后的重启次数 事件引起失败的比例 失败时数据崩溃的可能性





# 非功能需求的度量

- 非功能需求：检验起来非常困难，一般采用一些可度量的特性进行描述。
- 例如：
  - 即使对一个没有经验的用户，系统也应该很容易使用，且用户错误降到最少；
- 修改为：
  - 对一个没有经验的用户来说，经过**2**个小时的培训就应该使用系统的全部功能。在这样的培训之后，一个有经验的用户每天的出错平均数不应超过**2**次。



# 功能性需求 (FR) vs. 非功能性需求 (NFR)

- 非功能性需求
- [例]
  - 系统在**20**秒内响应所有的请求;
  - 系统应该每周**7**天、每天**24**小时都可使用;
- 注意: 非功能需求隐含了对可选设计方案的一些关键影响
  - 体系结构设计(e.g., 体系结构风格选择)
  - 算法设计(e.g., 排序策略的选择)



# 功能性需求 (FR) vs. 非功能性需求 (NFR)

## ■ 功能性需求 (FR)

- 描述了 **系统** 与其独立于系统实现 **环境** 之间的 **交互**。
- **环境** 包括用户和任何其他与该系统进行交互的外部系统。

## ■ 非功能性需求 (NFR)

- 描述了 **不直接关联** 到系统功能行为的系统的方方面面。从各个角度对系统进行约束和限制，反映了 **客户对软件系统质量和性能的额外要求**，如响应时间、数据精度、可靠性等。
- **可用性 (Usability)**: 是一种用户可以学会的操作、输入准备、解释一个系统或者构件输出的状况。
- **可靠性 (Reliability)**: 是系统或构件在给定时间内、指定条件下，完成其要求功能的能力。
- **性能 (Performance)**: 需求要考虑系统的定量属性，比如响应时间，吞吐量、有效性和准确性。
- **可支持性 (Supportability)**: 需求关注于在进行部署后系统的变化状况，比如包括可适配性、可维护性、可移植性等。

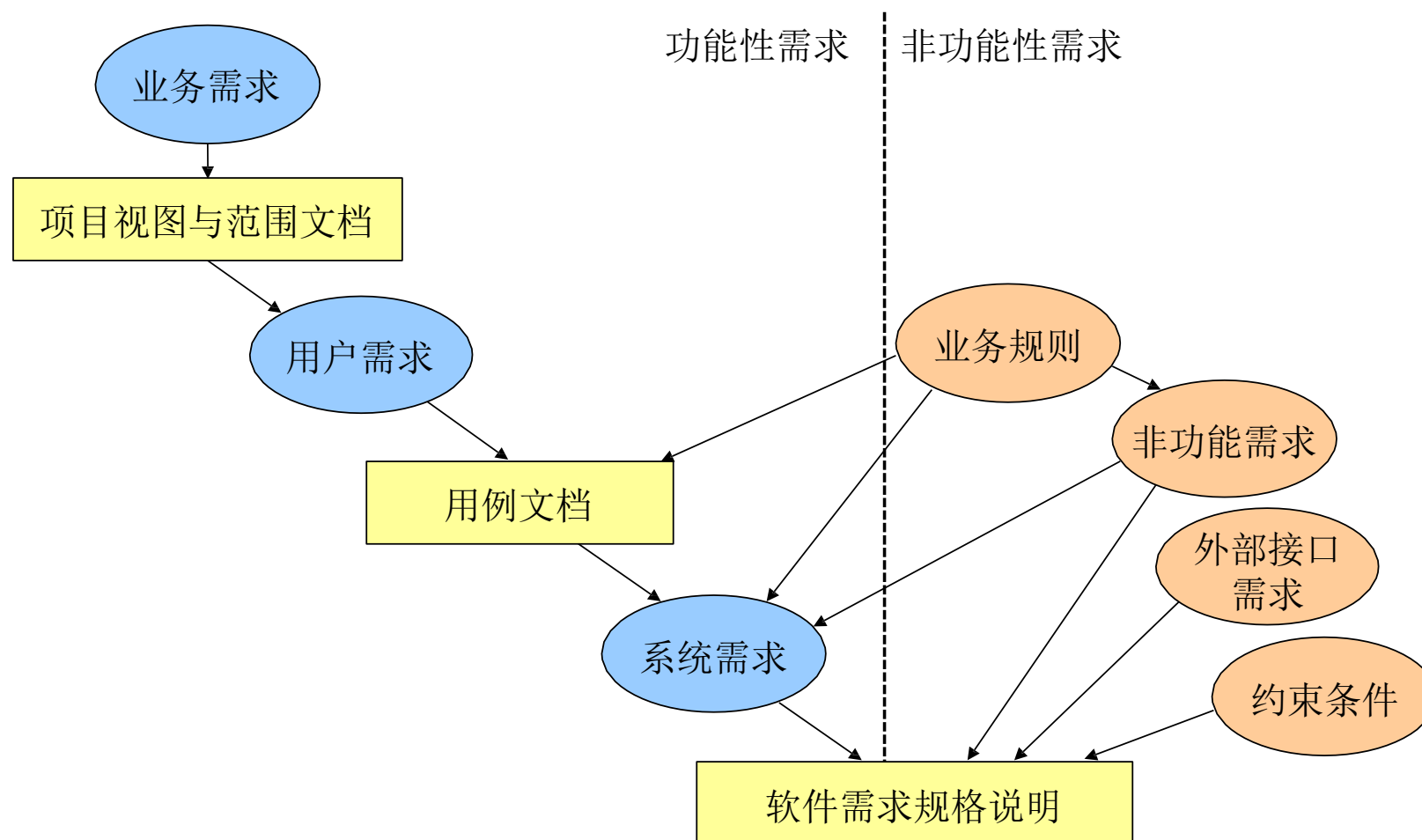


# 一个例子：拼写检查器

- 业务需求：“用户能有效地纠正文档中的拼写错误”
- 用户需求：“找出文档中的拼写错误并通过一个提供的替换项列表来供选择替换拼错的词”
- 功能性需求：
  - 找到拼写错误的单词并以高亮度提示
  - 显示提供替换词的对话框
  - 实现整个文档范围的替换
- 非功能性需求：
  - 正确的找到至少95%以上的错词并100%的加以正确替换
  - 拼写检查的速度应至少达到5000词/秒。



# 不同层次的软件需求





# 业务规则

- **业务规则(Business Rule):** 对某些功能的可执行性或内部执行逻辑的一些限定条件。
  - 通常表达为“如果..., 那么...”的形式
  - 通常是一些容易发生变化的功能;
- 例如:
  - 如果借书卡类型为“教师”, 那么一次借阅的最大数量为8本;
  - 如果订单金额大于10000元, 那么该订单的折扣为10%;
  - 如果采购单金额在10万到50万之间, 那么需要总经理审批;



# 外部接口需求

- **外部接口需求(External Interface Requirement):** 描述系统与其所处的外部环境之间如何进行交互，包括：
  - 硬件接口需求
  - 软件接口需求
  - 通信接口需求
- **例如:**
  - “从<某些设备>读取信号”
  - “给<一些其它系统>发送消息”
  - “以<某种格式>读取文件”
  - “能控制<一些硬件>”
  - “采用<某种类型的>用户界面”

**Exp1:** 账户接收系统必须为月财务状况系统部分提供更新信息，如在“财务系统描述”第4修订版中所描述的。

**Exp2:** 引擎控制系统必须正确处理从飞行控制系统接收来的命令，符合接口控制文档B2-10A4，修订版C的1到8段的规定



# 外部接口需求

外部接口分为以下几类：

- 1) **系统接口 (system interfaces)**：描述一个应用如何与系统的其他应用进行交互。
- 2) **用户接口 (user interfaces)**：规约了软件产品和用户之间接口的逻辑特性。即规约对给用户所显示的数据，对用户所要求的数据以及用户如何控制该用户接口。
- 3) **硬件接口 (hardware interfaces)**：如果软件系统必须与硬件设备进行交互，那么就应说明所要求的支持和协议类型。
- 4) **软件接口 (software interfaces)**：允许与其他软件产品进行交互，如，数据管理系统，操作系统或数学软件包。
- 5) **通讯接口 (communications interfaces)**：规约待开发系统与通讯设施（如，局域网）之间的交互。如果通讯需求包含了系统必须使用的网络类型（TCP/IP, WindowsNT），那么有关类型的信息就应包含在SRS中。





# 约束条件

- **约束条件 (Constraint)**：限制了开发人员设计和构建系统时的选择范围。
- 例如：
  - 成本、进度、技术选型、标准和规范等要求。
  - 如成本不能超出50万元，要求半年内交付使用等等。
  - 法律法规等。



# 好的需求应具备的特征

- **完整性**：每一项需求都必须将所要实现的功能描述清楚
- **正确性**：每一项需求都必须准确地陈述其要开发的功能；
- **可行性**：每一项需求都必须是在已知系统和环境的权能和限制范围内可以实施的
- **必要性**：每一项需求都应把客户真正所需要的和最终系统所需遵从的标准记录下来
- **划分优先级**：给每项需求、特性或使用实例分配一个实施优先级以指明它在特定产品中所占的分量
- **无二义性**：对所有需求说明的读者都只能有一个明确统一的解释
- **可验证性**：检查一下每项需求是否能通过设计测试用例或其它的验证方法，如用演示、检测等来确定产品是否确实按需求实现



# 产生不合格需求的原因

## ■ 无足够用户参与

- “我不明白为什么要花那么多功夫收集需求”
- “与其与用户讨论浪费时间，不如写代码有意思”
- “我已经明白用户需求了”

## ■ 用户需求的不断增加

- 若不断增加新需求，项目就越来越庞大以致超过其计划及预算范围
- 开发中不断延续的变更会使其整体结构日渐紊乱，补丁代码也使得整个程序难以理解和维护



# 产生不合格需求的原因

## ■ 模棱两可的需求

- 诸多读者对需求说明产生了不同的理解
- 单个读者能用不止一个方式来解释某个需求说明
- 后果：返工，重做一些你认为已做好的事情

## ■ 不必要的特性

- “画蛇添足”，开发人员力图增加一些“用户欣赏”但需求规格说明中并未涉及的新功能
- 客户可能要求一些看上去很“酷”，但缺乏实用价值的功能，而实现这些功能只能徒耗时间和成本



# 产生不合格需求的原因

## ■ 过于精简的规格说明

- 给开发人员带来挫折，使他们在不正确的假设前提和极其有限的指导下工作
- 给客户带来烦恼，他们无法得到他们所设想的产品

## ■ 忽略了用户分类

- 软件由不同的人使用其不同的特性
- 使用频繁程度有所差异
- 使用者受教育程度和经验水平也不尽相同

## ■ 不准确的计划

- 对需求分析缺乏理解会导致过分乐观的估计



## 3.1 软件需求与需求工程

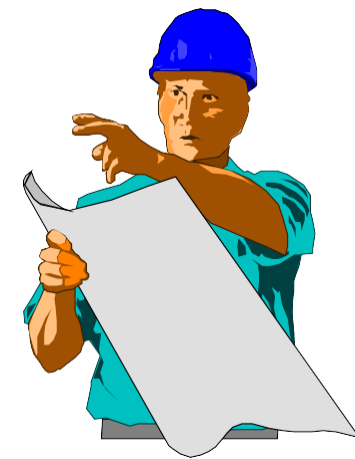
- 软件需求
  - 需求定义
  - 需求分类
- 需求工程



# 需求工程

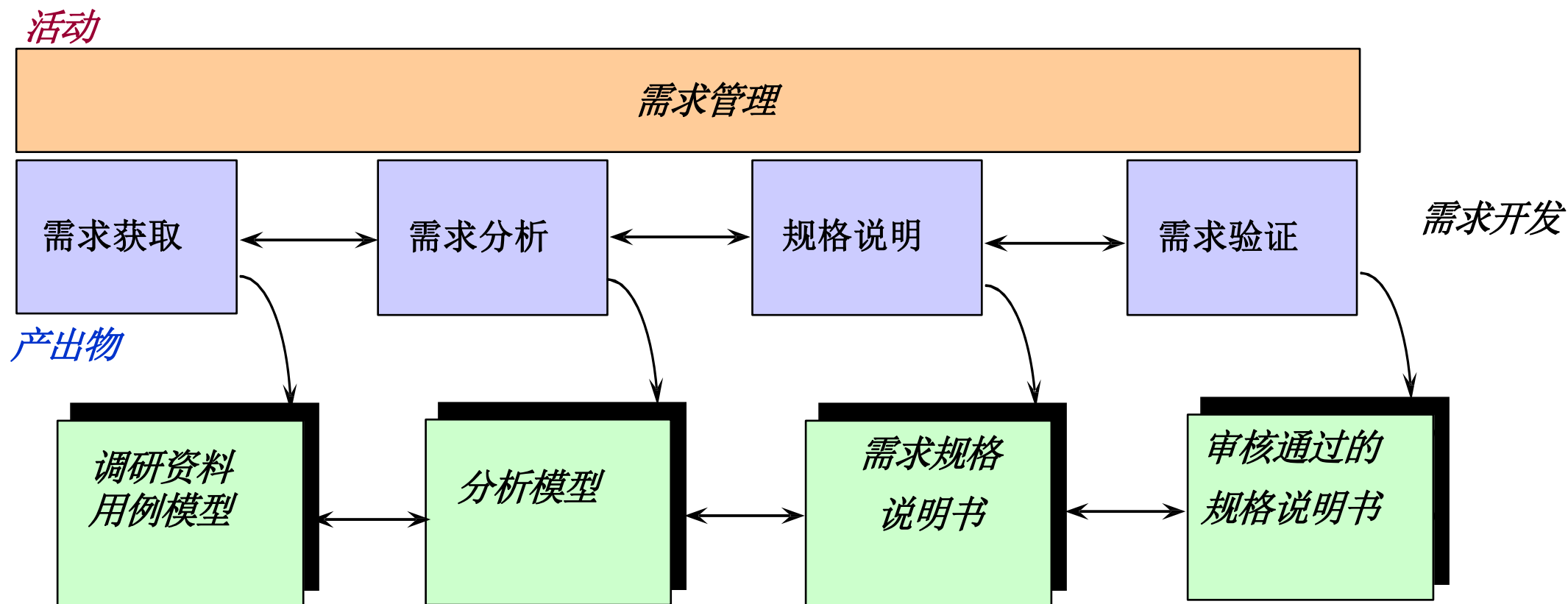
## ■ 需求工程(Requirement Engineering, RE)

- 应用已证实有效的技术、方法进行需求分析，**确定客户需求**，帮助分析人员理解问题并定义目标系统的所有外部特征的过程。
- 通过合适的工具和记号系统地描述待开发系统及其行为特征和相关约束，**形成需求文档**，并对用户不断变化的需求演进给予支持。
- 分析并记录软件需求，把**需求分解成一些主要的子系统和任务**，把这些子系统或任务分配给软件；通过一系列重复的分析、设计、比较研究、原型开发过程把这些系统需求转换成软件的需求描述和一些性能参数。





# 需求工程的总体流程







# 需求获取

- **需求获取(Requirement Elicitation):** 通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求
  - 对用户进行分类
  - 聆听每一类用户的需求
  - 分析和整理所获取的需求
  - 形成文档化的描述

# 需求获取

- **目标：**主动与干系人协同工作，找出他们的需求，识别潜在的冲突，磋商解决矛盾，系统边界的定义
- **实质：**了解待解决的问题及其所属领域
- **关键：**确保该问题的解决是有商业价值的





# 出发点

- 确定干系人 (stake holders)
  - 这里需要强调与客户之间的联络关系
  - 系统的设计到底与谁的利益息息相关
- 定义边界
  - 怎样界定问题的范围?
- 定义目标与情景实例
  - 目标与情景实例是组织原始需求信息的有效手段
- 分析可行性
  - 如何进行可行性研究
  - 如何选择好的项目
- 分析风险
  - 风险管理应长期、持续进行，而非阶段性、一次性的任务
  - 进行灾难及事故分析，以确定风险



# 干系人 (stake holders)

- 干系人分析
  - 找出所有干系人
  - 分析其隶属于哪个世界
- 干系人举例
  - **用户**—关心新系统特征和功能
  - **设计师**—想要构造完美的系统，尽量重用已有的代码
  - **系统分析师**—想要获取正确的需求
  - **培训与用户支持人员**—确保系统可用和可管理
  - **业务分析师**—想确保“我们做得比竞争对手好”
  - **技术文档作者**— 为系统准备用户手册及其他相关文档
  - **项目经理**—希望按时、按预算、按目标完成项目
  - **客户**—为新系统买单的人



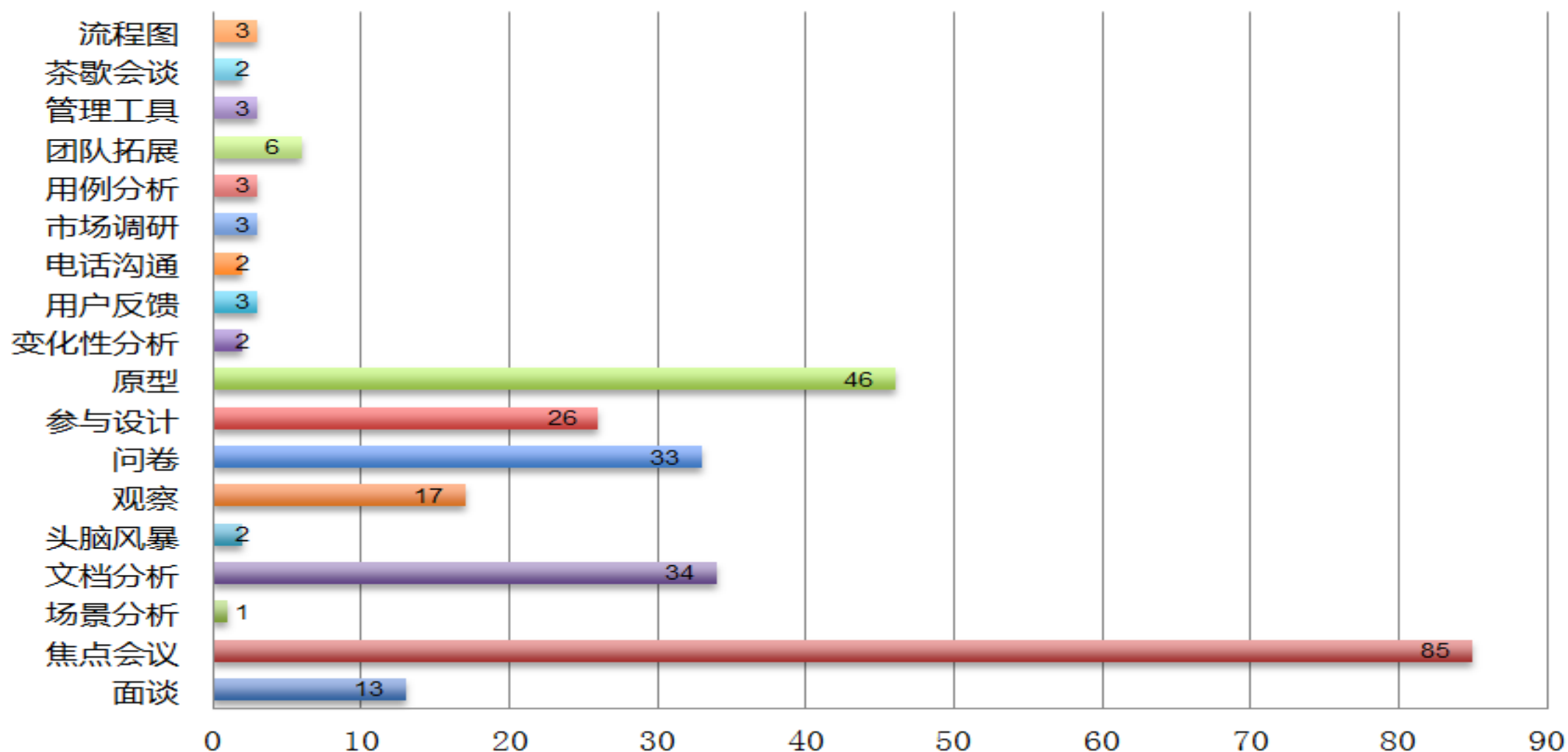
# 需求获取

- 获取技术
  - 协同工作 (Collaborative sessions)
  - 面谈 (Interviewing techniques)
  - 问卷调查 (Questionnaires)
  - 观察法 (Ethnography)
  - 原型法 (Prototyping)
  - 文档分析 (Documentation)
  - 建模 (Modeling)
  - 角色扮演 (Roleplaying)
  - 非功能性需求列表 (Checklists of NFRs)
- 冲突识别与磋商 (Conflict Identification and Negotiation)





## 软件需求获取方法应用现状调研结果



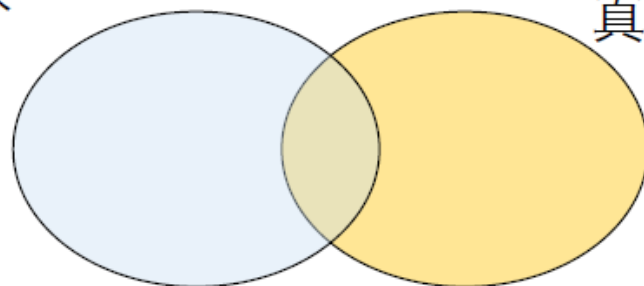




# 需求获取与用户参与

原始需求

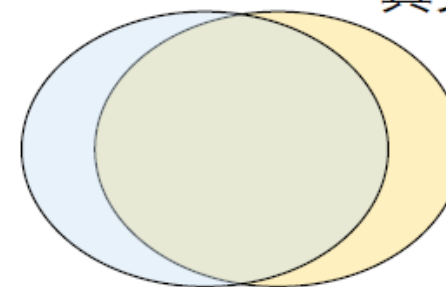
真实需求



无用户参与的需求获取结果

原始需求

真实需求



有用户参与的需求获取结果



# 需求分析

- **需求分析(Requirement Analysis)**: 对收集到的需求进行提炼、分析和审查, 为最终用户所看到的系统建立概念化的分析模型
  - 定义系统的边界
  - 建立软件原型
  - 分析需求可行性
  - 确定需求优先级
  - 建立需求分析模型
  - 创建数据字典





# 需求分析

- **目标：** 对产品及其与环境的交互进行更深入的了解，识别系统需求，设计软件体系结构，建立需求与体系结构组件间的关联，在体系结构设计实现过程中进一步识别矛盾冲突，并通过干系人之间的协调磋商解决问题。
- **实质：** 概念建模——选择常用的建模语言，进行功能建模和信息建模
- **关键：** 体系结构设计与需求分配

通过评估需求的满足度来评价体系结构设计的质量

# 需求分析两种方式

- 不要尝试向干系人证明你是聪明的
- 抓住所有的机会表现你认为干系人是聪明的
- 对比这两种情况



好的，告诉我为什么你觉得电梯慢

我们的电梯太慢了！

我不这么认为，我觉得你的电梯有吞吐量问题，不是速度问题



# 需求分析的两种方式



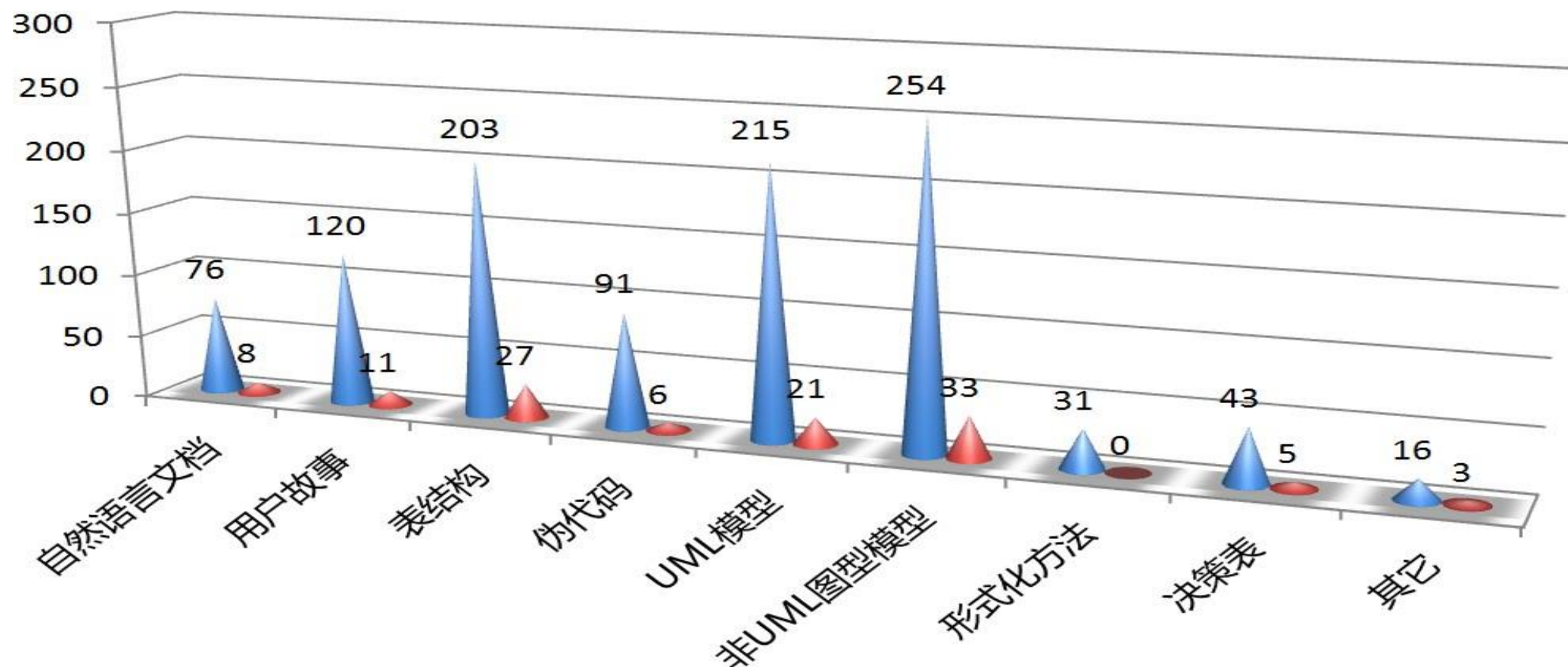
没有广告屏幕

有广告屏幕





# 需求建模方法





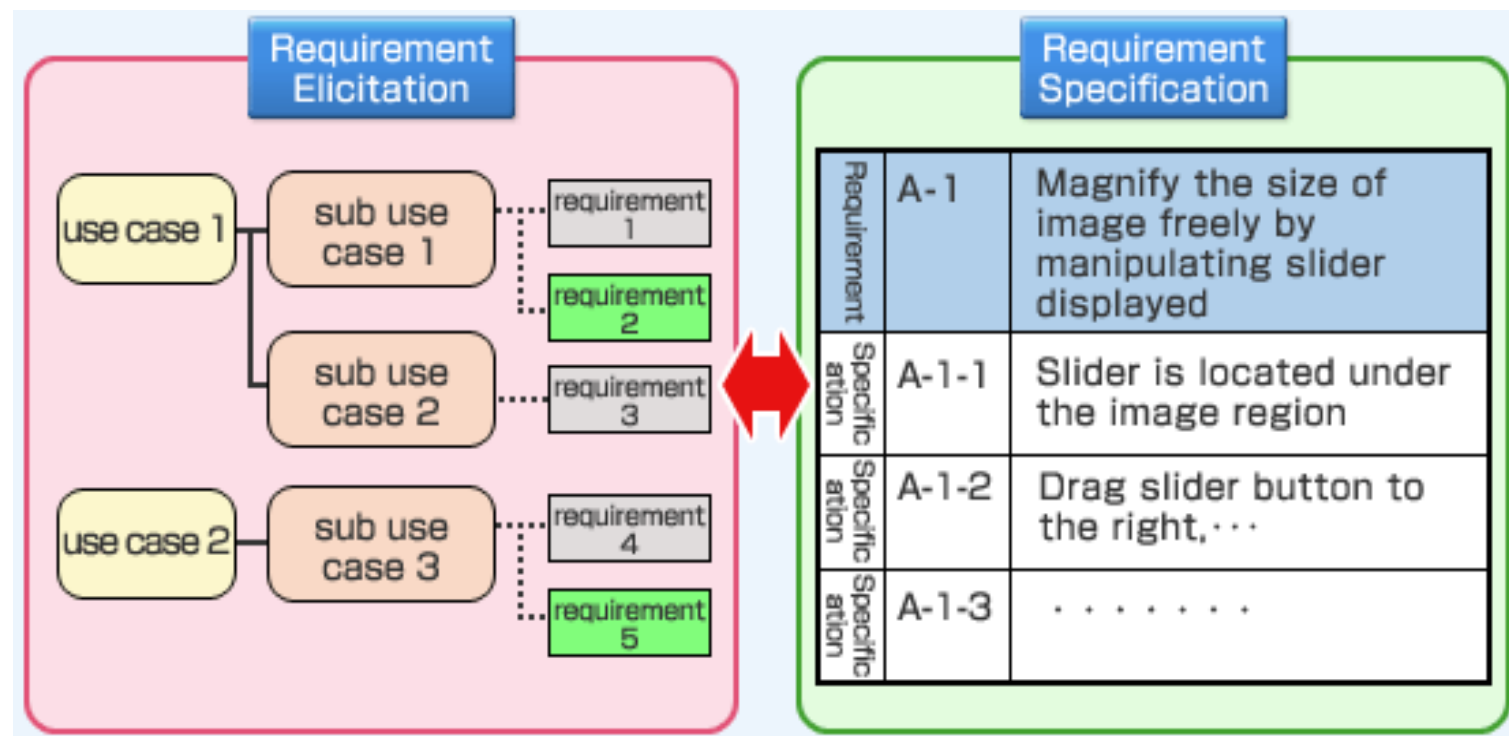
# 需求规格说明

- **需求规格说明(Software Requirement Specification, SRS):**
  - 需求开发的结果
  - 精确的、形式化的阐述一个软件系统必须提供的功能、非功能、所要考虑的限制条件等
  - 作为用户和开发者之间的一个契约
  - 是用户、分析人员和设计人员之间进行理解和交流的手段



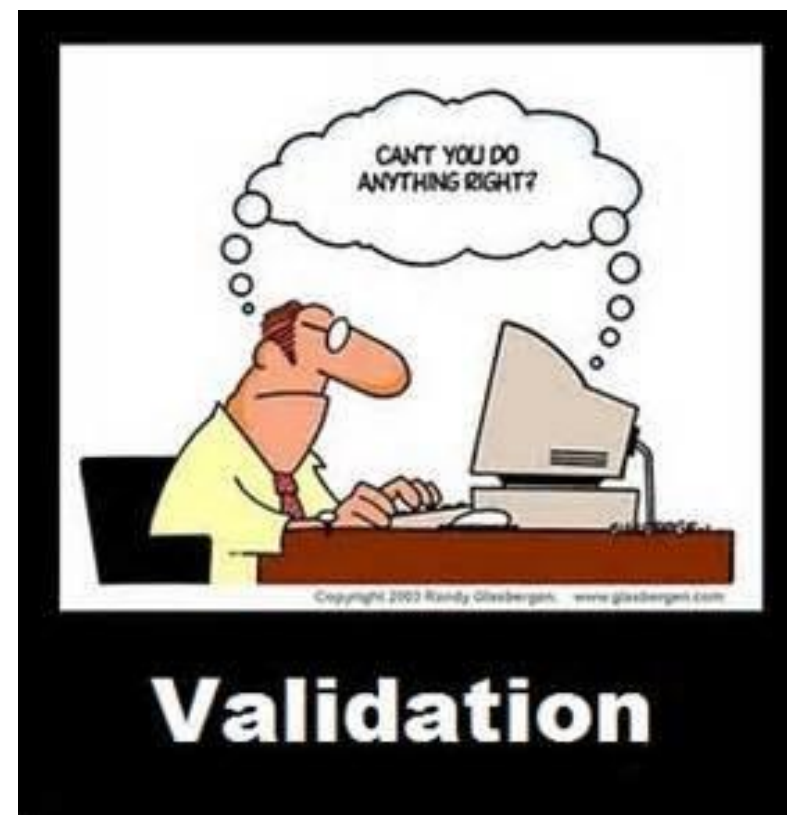
# 需求规格说明

- 系统与软件需求的文档化，以便于后续的需求及系统的正式评审，而准备的规范化文档。
- 单个需求项的质量
  - 准确 (Concise)
  - 正确 (Correct)
  - 明确 (Non-ambiguous)
  - 可行 (Feasible)
  - 可证 (Verifiable)
- 整个需求集合的质量
  - 现实 (Realistic)
  - 精确 (Concise)
  - 全面 (Complete)
  - 一致 (Consistent)



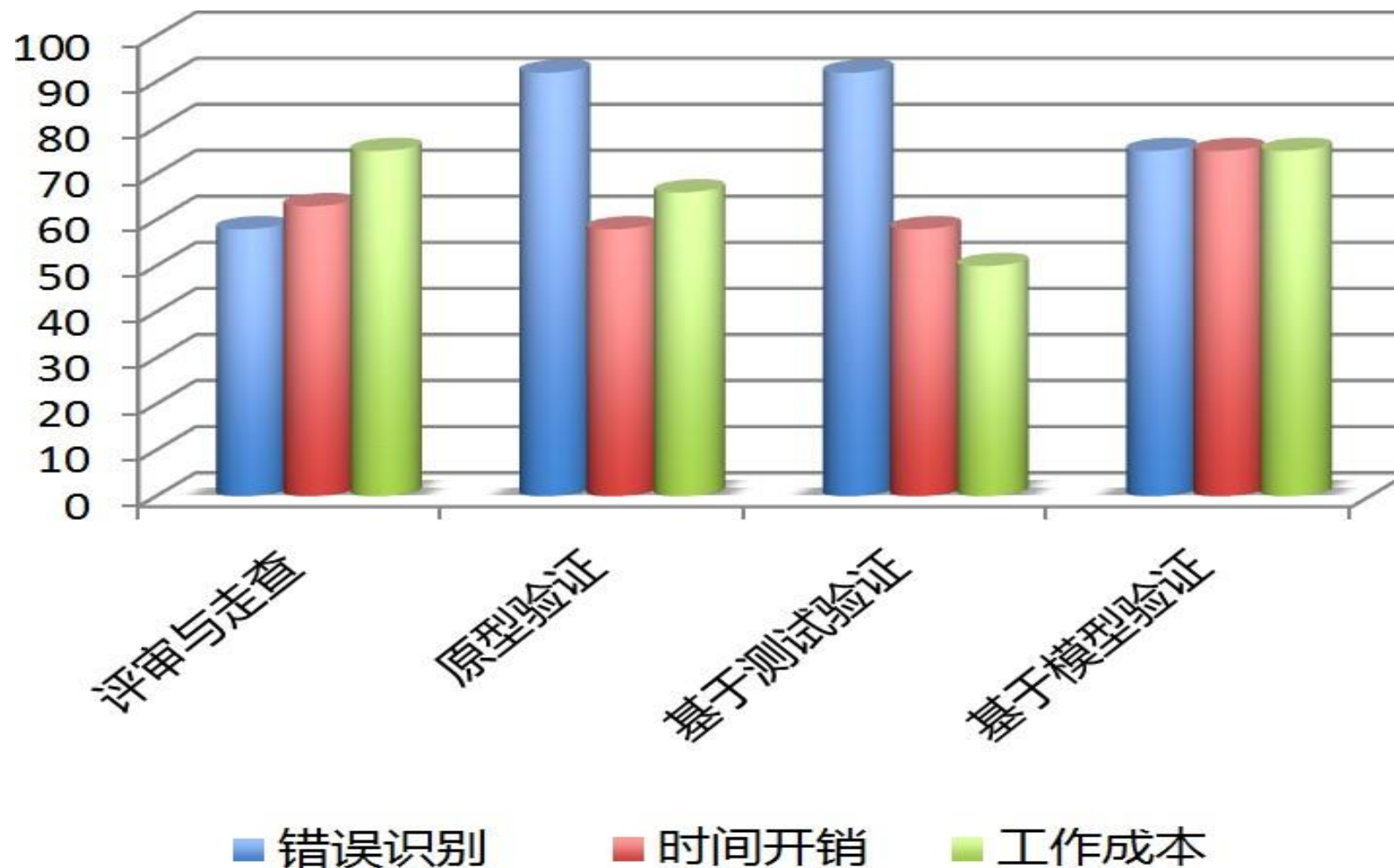
# 需求验证

- 对其他需求工程活动的质量的保证。通过数学的形式化工具或工程化的测试过程来确保系统满足干系人的要求。
- 验证方法
  - 评审 (Review)
  - 原型化 (Prototyping)
  - 模型验证 (Model validation)
  - 确认测试 (Acceptance Tests)





# 需求验证方法







# 需求管理

贯穿从需求获取到软件系统下线的全过程。需求管理涉及软件配置管理、需求跟踪、影响分析和版本控制

- 需求跟踪 (Requirements traceability)

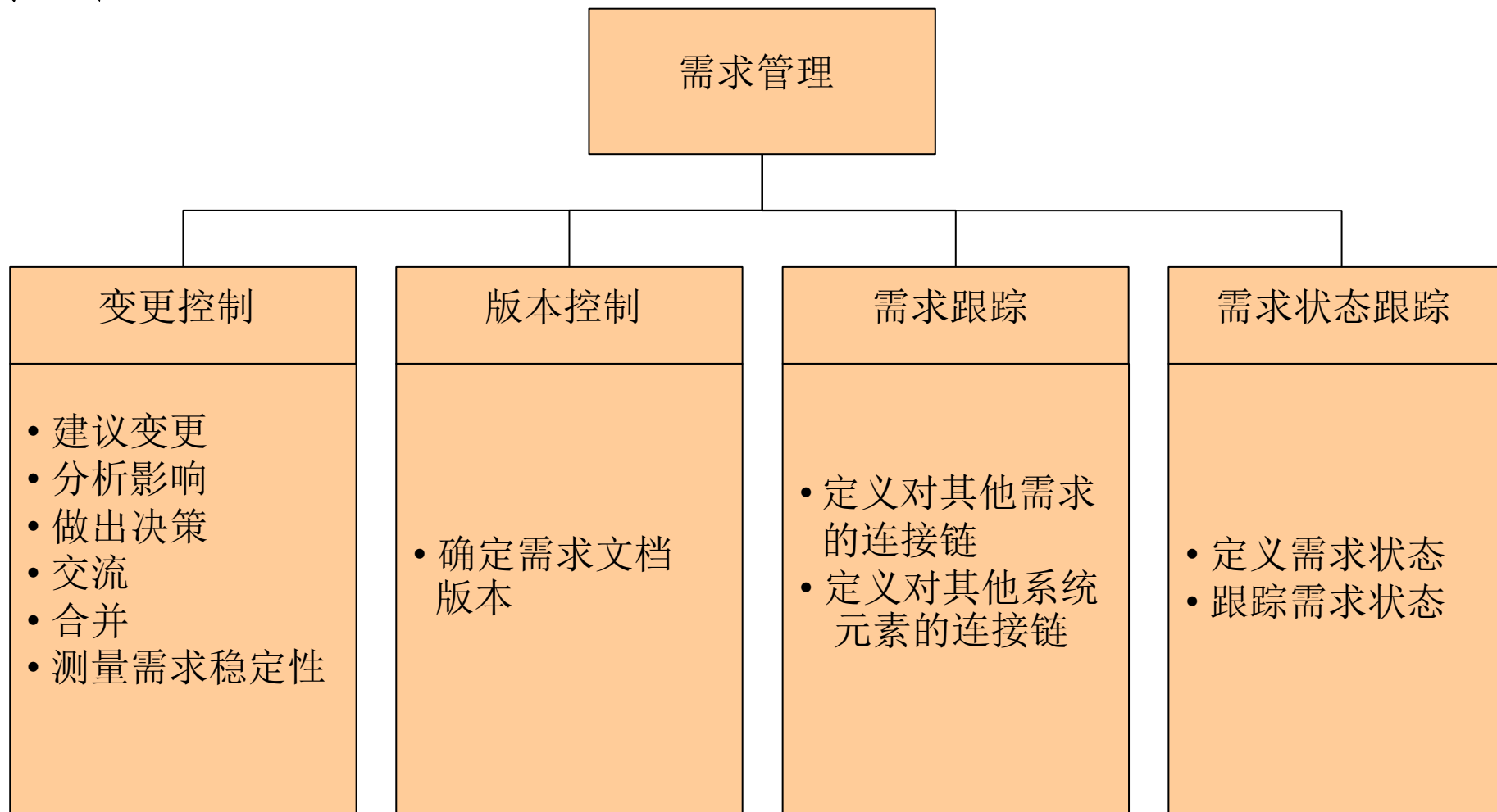
描述和追踪一条需求的来龙去脉的能力，包括向前追踪到软件制品，向后追踪到需求来源

- 变更请求管理 (Change Requests) 系统化的变更管理

- 需求属性管理 (Requirements attributes)



# 需求管理





# 小结

- 什么是软件需求
  - 软件需求的分类
  - 需求工程的概念
- 
- 下一节：需求获取与建模