5-1 软件需求与需求获取

2019年5月8日 22:01

需求工程师

- 软件工程师做出尽可能简化问题复杂度的假设
- 计算机科学家要找出不失一般性的解决方法
- 数学家追求对问题描述的精确性

当代的需求工程师

- 分析问题和解决问题的能力
- 人际沟通及交流能力
- 软件工程知识和技能
- 应用领域有关知识
- 书面语言组织和表达能力

优秀需求工程师的目标

- 识别错误假设
- 确保一致性
- 提升依从性
- 减少彼此误解
- 提高支持速度和效率
- 提升客户满意度
- 撰写优质需求文档

主要内容

1 软件需求的定义

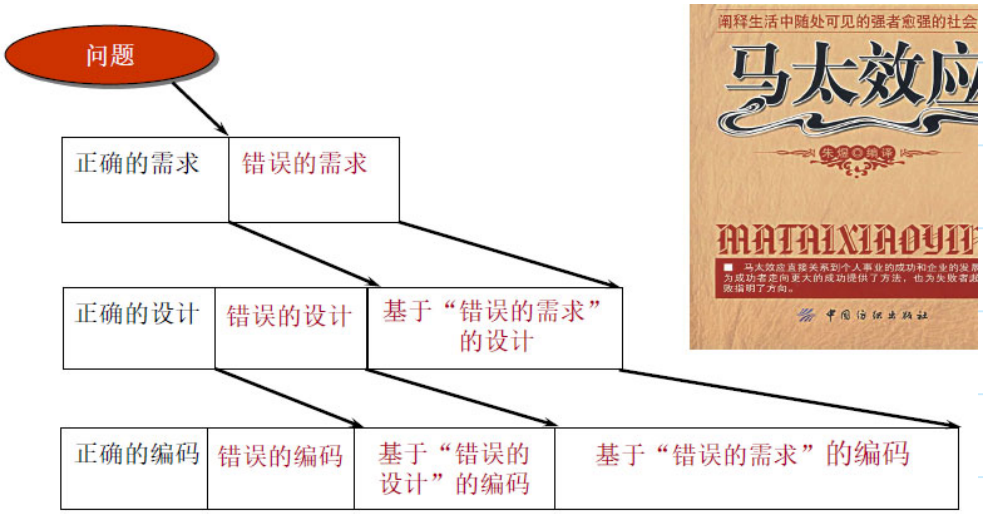
软件开发的目标是什么

- 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；
- 软件要能够满足顾客的需求。

但实际情况是什么样子？

- 调查报告的数字是这样的...

“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?

- 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。

什么是“软件需求”

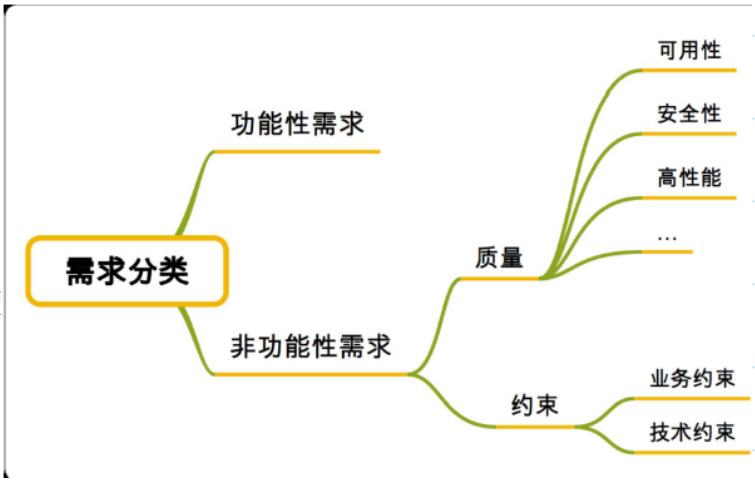
- 软件需求(Software Requirements)：
  - 用户解决问题以达到特定目标所需的能力；
  - 系统或系统构件要满足的合同、标准、规范或其他正式文档所需具备的能力； ——IEEE, 1997
- 软件需求：以一种清晰、简洁、一致且无二义性的方式，描述用户对目标软件系统在功能、行为、性能、设计约束等方面的期望，是在开发过程中对系统的约束。
- 需求通常用于表达“做什么”，而不描述“如何做”。

“软件需求”的作用

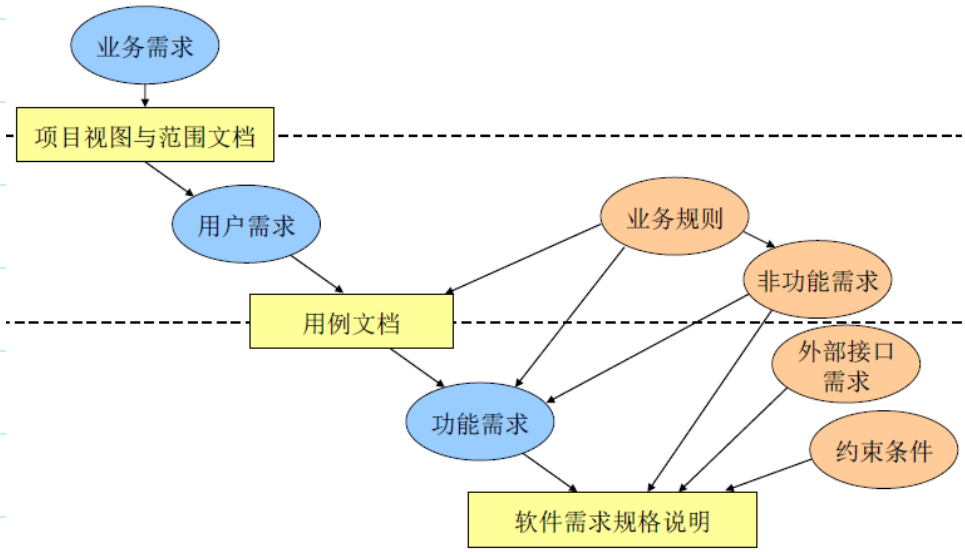
- 充分理解现实中的业务问题，并作为软件设计的基础；
- 为软件项目的成本、时间、风险估计提供准确的依据；
- 减少开发工作量，避免将时间与资源浪费在设计与实现错误的需求上
- 通过提供需求文档和需求基线，来有效的管理系统演化与变更；
- 作为顾客与开发团队之间正式合同的一部分；
- 为最终的验收测试提供标准和依据；

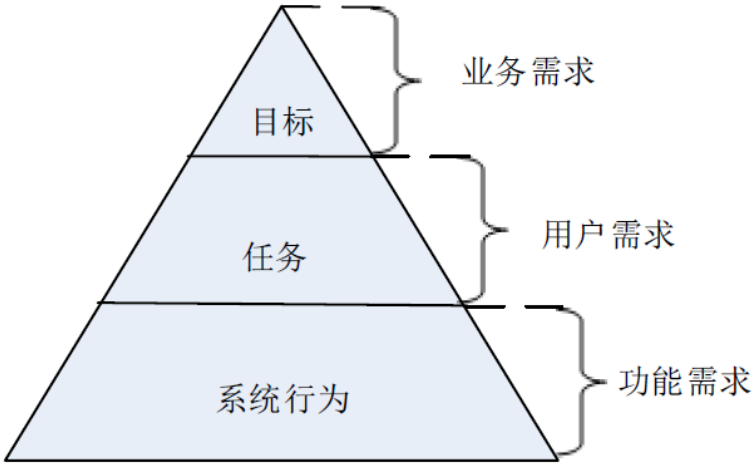
2 软件需求的分类

- 产品/过程
  - 产品需求
  - 过程需求
- 产品需求
  - 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



不同层次的软件需求





业务需求(Business Requirements)

- 客户对于系统的高层次目标要求(highlevelobjectives)，定义了项目的远景和范畴(visionandscope)
  - 业务：属于哪类业务范畴？应完成什么功能？为何目的？
  - 客户：软件为谁服务？目标客户是谁？

    - 特性：区别于其他竞争产品的特性是什么?
    - 价值：价值体现在那些方面?
    - 优先级：功能特性的优先级次序是什么?
- [例]“图书资料管理系统”的业务需求
    - 该系统使用计算机实现图书资料的日常管理，提高工作效率和服务质量
    - 该系统可让用户在网络上查询与浏览电子资料，改变原有的借阅模式
    - 由于版权的限制，某些电子资料只能浏览/打印，但不能下载。

用户需求(User Requirements)

- 从用户角度描述的系统功能需求与非功能需求，通常只涉及系统的外部行为而不涉及内部特性。
- [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；
    - [功能需求]用户通过Internet查询图书信息；
    - [功能需求]用户通过Internet浏览个人借阅情况；
    - [功能需求]用户通过Internet查找和浏览电子资料；
    - [非功能需求]随时、快速

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，学校领导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。

功能需求(Functional Requirements)

- 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
- [例]
    - 用户可从图书资料库中查询或者选择其中一个子集；
    - 系统可提供适当的浏览器供用户阅读馆藏文献；
    - 用户每次借阅图书应对应一个唯一的标识号，它被记录到用户的账户上

功能需求的层次性(F)

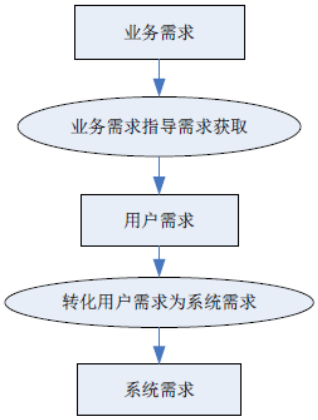
- 将用户需求转化为功能需求的过程是一个复杂的过程
  - 首先需要分析问题领域及其特性，从中发现问题域和计算机系统的共享知识，建立系统的知识模型；
  - 然后将用户需求部署到系统模型当中，即定义系列的系统行为，让它们联合起来实现用户需求，每一个系统行为即为一个系统需求。
  - 该过程就是需求工程当中最为重要的需求分析活动，又称建模与分析活动。
- 

图　不同抽象层次需求之间的联系

非功能性需求(Non‐Functional Requirements)

- 从各个角度对系统的约束和限制，反映了客户对软件系统质量和性能(qualityandperformance)的额外要求，如响应时间、数据精度、可靠性等。
- [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；
  - 对一个没有经验的用户而言，经过2小时培训即可使用系统的所有功能。
- 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(eg,体系结构风格选择)
  - 算法设计(eg,排序策略的选择)
- 非功能性需求(URPS+)
  - 描述了不直接关联到系统功能行为的系统的方方面面。从各个角度对系统的约束和限制，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。
  - 其他需求(+)
    - 实现需求：资源限制、语言和工具、硬件等

        - 界面需求：人机交互
        - 操作需求：对其操作设置的系统管理
        - 打包需求：发布形式
        - 合法需求：应用其他软件的许可等

非功能需求的度量

- NFR：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，且使用户错误降到最少；
- 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。
-

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	内存空间数 硬盘空间数
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

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。

约束条件(Constraints)

- 系统设计和实现时必须满足的限制条件，对其进行权衡或调整是相当困难的，甚至是不可能的；

- 例如：
  - 系统必须用C++或其他面向对象语言编写；
  - 系统用户接口需要采用图形化界面；
  - 任取10秒，一个特定应用所消耗的可用计算能力平均不超过50%；
  - 系统开发过程和交付文档需遵循GB/T8567－2006标准；
  - 通讯接口必须符合ISO七层架构。
- 来源：法规政策、硬件/资源限制、开发语言、等等（依从性需求）。

业务规则

- 业务规则(BusinessRule)：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(ExternalInterfaceRequirement)：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”

关于需求的一些例子

- 系统必须有能力支持100个以上的并发用户，每个用户可以处理操作任务的任选组合，平均响应时间应该小于1秒，最大响应时间应小于5秒
- 必须在对话窗口的中间显示错误警告，使用红色的、14点加粗Arial字体
- 系统必须有能力存储平均操作连续100天所产生的事务。
- 系统应该在5分钟内计算出给定季度的总销售税。

- 系统应该在1分钟内从1000000条记录中检索出一个销售订单。
- 系统必须支持100个Windows工作站的并行访问。
- 系统可从各型号的modem上读取信号作为系统输入。

3 好的需求vs坏的需求

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
- 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
- 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
    - 原因：频繁的需求变更、遗漏的需求、与用户交流不够、质量低下的需求规格说明和不完善的需求分析

考虑以下需求是否满足“好需求”的标准，如不是，该如何修正?

    - 1在用户每次存钱时系统将进行信用检查;
    - 2如果用户试图透支，系统将采取适当的行动;
    - 3系统将尽可能快的响应所有有效的请求;
    - 4系统允许立即使用所存资金;
    - 5只有在手工验证所存资金后，系统才能允许使用

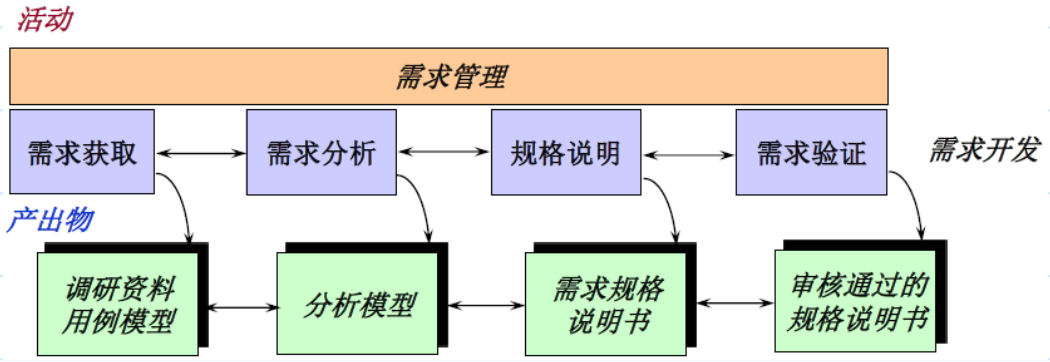
答

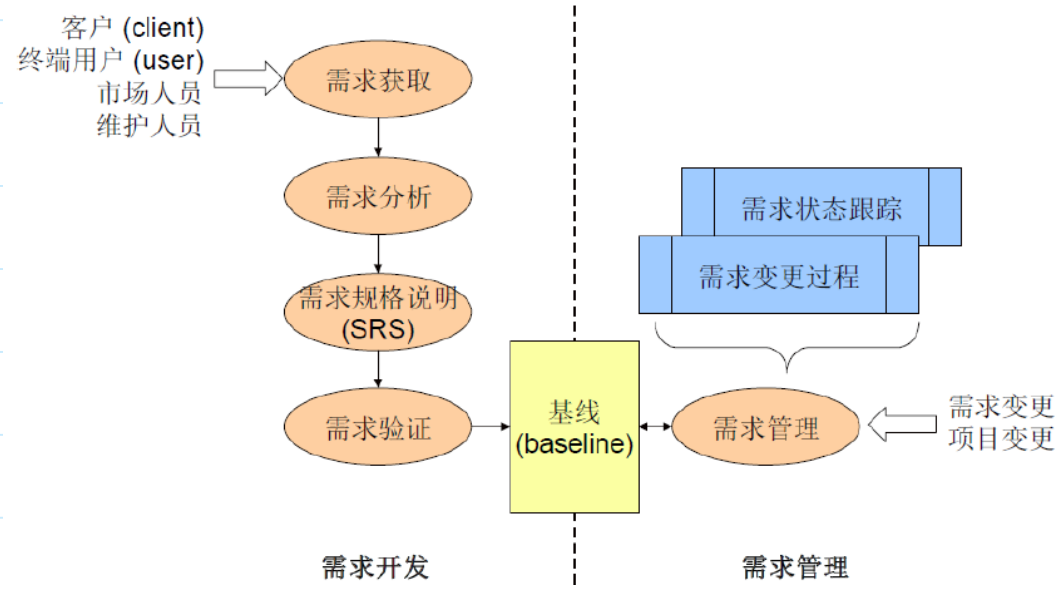
    - 1在现实情况中，用户存钱时并不需要信用检查，因此这个需求描述是错误的
    - 2“适当的行动”对不同的人来说有不同的解释，显然是歧义的。改正：如果用户试图透支，系统将显示错误信息并拒绝取款操作。
    - 3“尽快”是不可验证的，应该给出具体数量值。改正：系统将在20秒内响应所有有效的请求。
    - 4与5是矛盾的。

4 需求工程(RequirementEngineering,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

- 

- 

需求开发所包含的活动

- 确定产品所期望的用户类
- 获取每个用户类的需求
- 了解实际用户任务和目标以及这些任务所支持的业务需求
- 分析源于用户的信息以区别用户需求、功能需求、非功能需求、约束条件、建议解决方法和附加信息
- 将系统级的需求分为几个子系统，并将需求中的一部份分配给软件构件
- 了解相关非功能属性的重要性
- 商讨实施优先级的划分
- 将所收集的用户需求编写成规格说明和模型
- 评审需求规格说明，确保对用户需求达到共同的理解与认识，并在整个开发小组接受说明之前将问题都弄清楚

(1)需求获取(RequirementElicitation)

- 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
    - 签字确认
- 抽取技术
    - 协同工作（Collaborativesessions）
    - 面谈（Interviewingtechniques）
    - 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（ChecklistsofNFRs）
- 冲突识别与磋商（ConflictIdentificationandNegotiation）

(2)需求分析(RequirementAnalysis)

- 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
    - 设计软件体系结构
    - 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

(3)需求规格说明、(4)需求验证

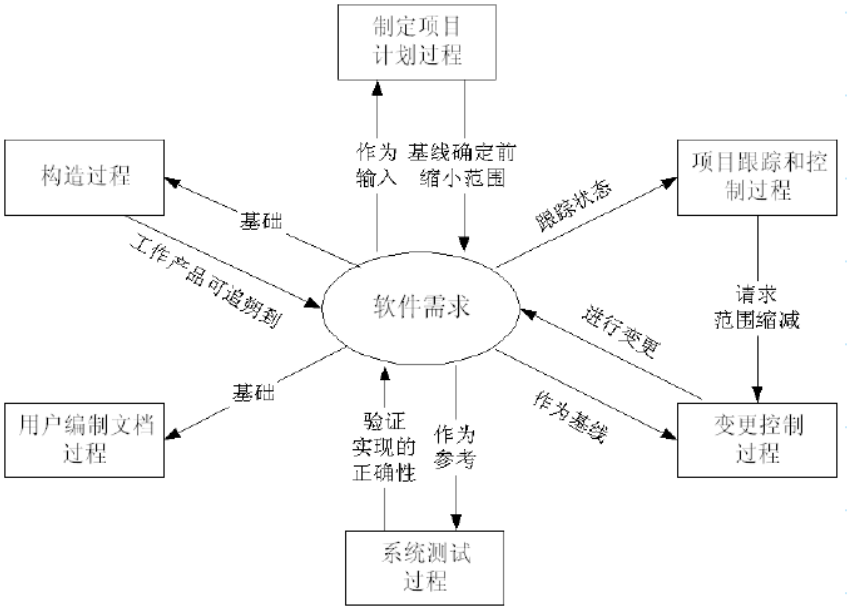
- 需求规格说明(SoftwareRequirementSpecification,SRS)：
    - 需求开发的结果
    - 精确的、形式化的阐述一个软件系统必须提供的功能、非功能、所要考虑的限制条件等
    - 作为用户和开发者之间的一个契约
    - 是用户、分析人员和设计人员之间进行理解和交流的手段
- 需求验证(RequirementVerification)：以需求规格说明为输入，通过评审、模拟或快速原型等途径，分析需求规格的正确性和可行性，发现存在的错误或缺陷并及时更改和补充。

(5)需求管理(RequirementManagement)

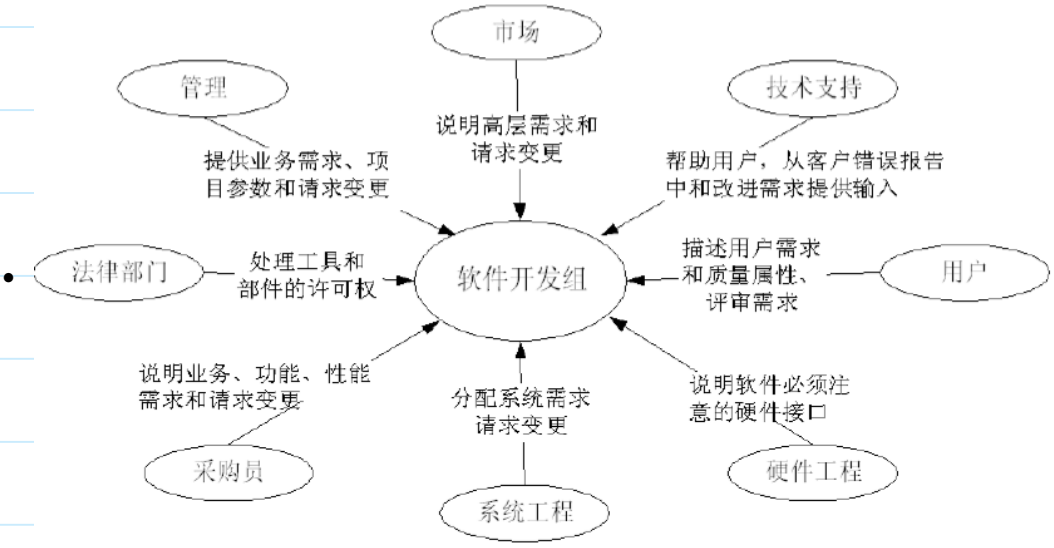
- 定义需求基线(迅速制定需求文档的主体)

- 评审提出的需求变更、评估每项变更的可能影响从而决定是否实施它
- 以一种可控制的方式将需求变更融入到项目中
- 使当前的项目计划与需求一致
- 估计变更需求所产生影响并在此基础上协商新的承诺(约定)
- 让每项需求都能与其对应的设计、源代码和测试用例联系起来以实现跟踪
- 在整个项目过程中跟踪需求状态及其变更情况

5 *软件需求与SE其他要素的关系



软件需求对其他项目成员的影响



6 *与需求有关的风险

需求获取的风险

- 产品视图与范围可能不断扩大

- 需求分析所需时间可能会被压缩
- 需求规格说明的完整性和正确性难以保证
- 非功能需求可能被忽略
- 可能只考虑了部分用户而疏漏了其他用户的需求
- 客户可能会有一些隐含的期望需求，但未加说明
- 把已有的产品作为需求基线
- 用户当前所用的解决方法往往掩盖了用户的实际需求

需求分析与需求规格说明的风险

- 需求分析
  - 不恰当的划分需求优先级划分
  - 带来技术困难的特性
  - 不熟悉的技术、方法、语言、工具或硬件平台
- 需求规格说明
  - 开发人员和客户对需求的不同理解会带来彼此间的期望差异
  - 时间压力对TBD的影响
  - 具有二义性的术语
  - 需求说明中包括了设计

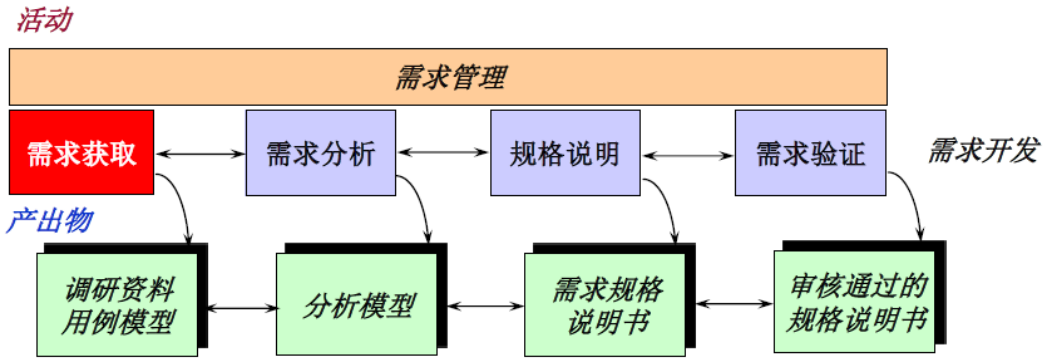
需求验证与需求管理的风险

- 需求验证
  - 未经验证的需求
  - 审查的有效性无法保证
- 需求管理
  - 变更需求
  - 不按计划的过程来做出变更
  - 未实现的需求
  - 扩充项目范围

7 需求获取

- 7.3 需求获取的主要内容

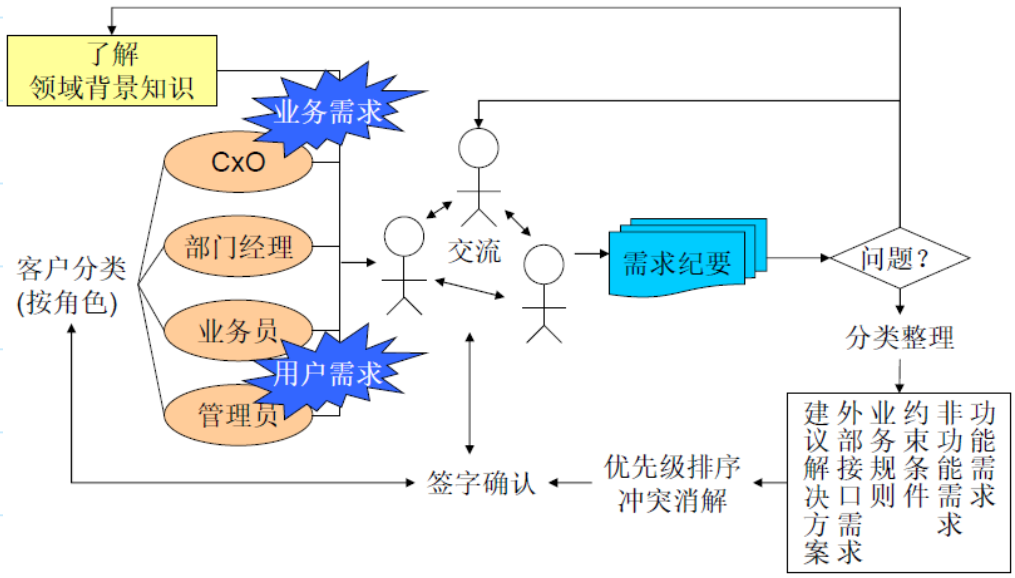
需求工程的总体流程

- 

需求获取的目标

- 收集准备建立的系统和正在使用的系统的信息，并从这些信息中提取用户和系统需求。
- 为下一步的需求分析提供素材。

需求获取的基本步骤

- 

1. 了解相关背景和领域/行业的知识，确定产品所期望的用户类；
2. 与客户企业或组织的高层人员进行交流，了解实际用户任务和目标以及这些任务所支持的业务需求；
3. 与客户企业或组织的底层人员进行交流，获取每个用户类的详细的用户需求；
4. 整理需求纪要，发现新问题，并重复1 - 3步；
5. 需求分类和组织，以区别功能需求、非功能需求、约束条件、业务规则、外部接口需求、建议解决方法和附加信息；
6. 优先排序和冲突解决；
7. 得到最终需求清单，并与客户做最终签字确认。

7.1 需求获取所面临的挑战

- “Yes,But”综合症
  - “Yes,But”是软件开发中最经常遇到的问题，这种反应实际上是人的一种自然反应。
    - 不管之前他多么认同你的设计，在没有看到真正的系统之前，用户决不可能完全理解你的设计；——软件本质上的“无形性”造成的必然结果
    - 机械设计里的每一步都是看的见摸得到的，用户从最开始就能与设计人员同步理解，所以不存在“Yes,But”问题
    - 在软件设计里，需求获取阶段的一个重要目标就是如何尽早的把“But”后面的部分发现出来
- “Undiscovered Ruins”综合症
  - “我知道的就这些，你们还想知道什么？”
  - “我们已经发现了所有的需求，现在让我们开始着手开发吧...”
  - “知道的越多，不知道的也越多”
  - “UndiscoveredRuins”：请问尚未被发现的废墟有多少呢?
  - 需求是永无止境的，变化的!
- “UserandDeveloper”综合症
  - 软件开发中，开发人员与用户处于不同的知识、技术层面，所关注的目标不同，双方在沟通时必然存在communicationgap(交流的鸿沟)
    - “理解用户需求”这一目标驱使软件开发人员从他们所沉溺的“01”世界转入到现实中的世界;
    - 巨大的鸿沟(gap)导致开发人员与用户之间无法充分的相互理解;
    - 为了在两个截然不同的世界之间架起一座桥梁，有必要学习一些技术以便于有效的获取和理解客户需求。
- 小结

问题	解决方案
“Yes, But”综合症：直到开发人员将用户描述的东西交给他们，用户才认为他们知道自己要什么	尽早提供可选择的启发技术：**原型开发、迭代方法**等
“Undiscovered Ruins”综合症：用户不知道自己需要什么，或知道但不知如何表达	**详细调研**，将用户当作领域专家来认识和激励，尝试其他交流和启发技术
“User and Developer”综合症：分析员认为自己比用户更了解用户的需求	熟悉应用领域，把分析员放在用户的位置上，采用**用例分析方法**

7.2 需求获取技术

需求获取的途径

- 需求获取的关键：沟通和交流
- 所要避免的问题：交流障碍、沟通不全、意见冲突
- 所要必备的条件：较高的技术水平、丰富的实践经验、较强的人际交往能力
- 可能采取的手段：
  - 7.2.1 面对面访谈
  - 7.2.2 需求研讨会
  - 7.2.3 现场观察/体验
  - 7.2.4 头脑风暴
- 重点注意业务单据等资料的收集、整理！！--分析的依据
- 多种方法要复合在一起使用，效果更好

7.2.1 面对面访谈

- 需求获取中最直接的方法：用户面谈(interviewing)
- “看起来很美”，但“做起来并不容易”
- 需求分析者个人的偏见、事先的理解、以往的经验积累是导致面谈失败的最重要原因
- 在面谈时，忘掉一切以往所作的事情，通过问题启发，倾听对方的陈述
- 不要把自己放在“专家”的位置上

如何提问?

- “每个人都能提问题，但并不等于人人都会提问题...”
- 封闭式问题
  - 对错判断或多项选择题，回答只需要一两个词
- 开放式问题
  - 这种问题需要解释和说明，同时向对方表示你对他们说的话很感兴趣，还想了解更多的内容。
- 通过提问题增强你对谈话进展和方向的控制
- 问题不能过于宽泛
- 最开始的问题不能太难
- 不能在提问之前就已经表示不赞同
- 谈话之前有意识的准备一些备用问题
- Questioningskills：一般以开放式问题开始谈话，发现问题；以选择式问题引导；以封闭式问题确认事实；以探究式问题(probing，除了还有吗？)发现真实想法。

访谈问题的分类

- 上下文无关的问题(contextfreequestions)：充分理解用户的问题，不涉及具体的解决方案

    - 客户是谁?
    - 最终用户是谁?
    - 不同用户的需求是否不同?
    - 这种需求目前的解决方案是什么?
- 解决方案相关的问题(solutioncontextquestions)：通过这类问题，探寻特定的解决方案并得到用户认可
    - 你希望如何解决这个问题?
    - 你觉得该问题这样解决如何?

面谈之前

- 确立面谈目的
- 确定要包括的相关用户
- 确定参加会议的项目小组成员
- 建立要讨论的问题和要点列表
- 复查有关文档和资料
- 确立时间和地点
- 通知所有参加者有关会议的目的、时间和地点

面谈之中

1. 事先准备一系列上下文无关的问题，并将其记录下来以便面谈时参考
2. 面谈前，了解一下要面谈的客户公司的背景资料，不要选择自己能回答的问题而浪费时间;
3. 面谈过程中，参考事先准备的面谈模板，以保证提出的问题是正确的。将答案记录到纸面上，并指出和记录下未回答条目和未解决问题
4. 面谈之后，分析总结面谈记录。

面谈之后

- 复查笔记的准确性、完整性和可理解性
- 把所收集的信息转化为适当的模型和文档
- 确定需要进一步澄清的问题域
- 向参加会议的每一个人发出此次面谈的minutes(会议纪要)。

面谈记录的示例(1)

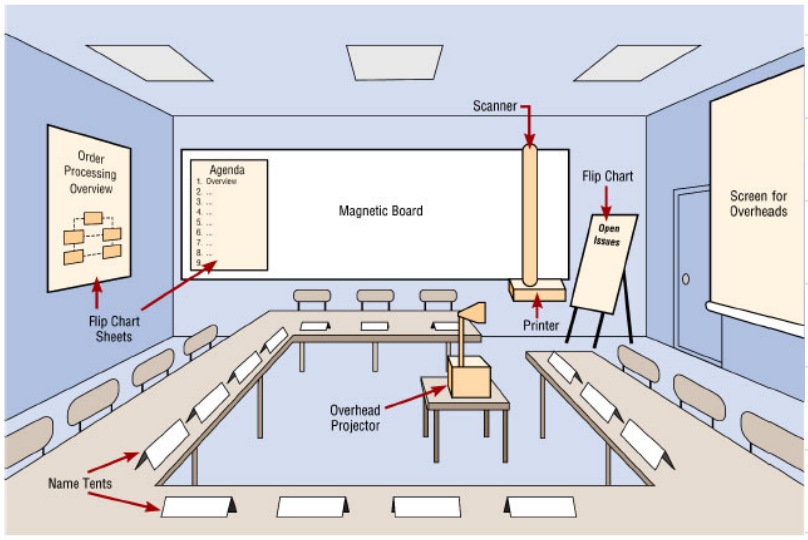
- 第一部分：建立客户或用户情况表
- 第二部分：评估问题
    - 询问用户对哪些类型的问题缺乏好的解决方案
    - 它们是什么？(不断的问“还有吗？”)

- 第三部分：理解用户环境
  - 谁是用户？他们的经历和经验如何？用户的预期如何？
- 第四部分：扼要说明理解情况
  - 你刚才告诉我：(用自己的话复述客户描述的问题)
  - 这是否足以表达你现在的解决方案中存在的问题？
  - 如果有，你还有什么问题？
- 第五部分：分析人员对客户问题的输入
  - 对每个问题进行以下提问：
    - 这是一个实际的问题吗？
    - 问题产生的原因是什么？
    - 现在如何解决的？
    - 希望如何解决？
    - 该问题的重要度如何？
- 第六部分：评估自己的解决方案
  - 总结自己建议的解决方案；
  - 对自己方案的优先级排序；
- 第七部分：评估机会
- 第八部分：评估可靠性、性能及其他需要
- 第九部分：其他需求
  - 法律法规、环境、行业标准等；
- 第十部分：总结性提问
  - 还有其他问题要问面谈人吗？
  - 尚未解决的问题有哪些？
  - 下次访谈的方式、地点、时间、参加人等；
- 第十一部分：分析人员的总结
  - 总结出客户/用户确认的三条优先级最高的需求或问题。

面对面访谈的优缺点分析

- 优点：
  - 人们很愿意谈论自己的工作，并且总是很喜欢接受访谈；
- 缺点：
  - 大多数人都采用专业术语和“行话”，而太多的专业术语让需求工程师难以理解，往往造成很多误解；
  - 有些需求对用户来说太普通了，以至于他们不自觉地认为这些需求太基本，不值得去提。但它们对需求工程师来说却不是显而易见的。这往往会造成某些需求被忽略；

7.2.2 需求研讨会

- 

- 通过让所有相关人员一起参加某个单一会议来定义需求或设计系统，也称联合应用设计会议(JointApplicationDesign,JAD)。
- 系统相关者在短暂而紧凑的时间段内集中在一起，一般为1至2天，与会者可以在应用需求上达成共识、对操作过程尽快取得统一意见。
- 协助建立一支高效团队，围绕一个目的：项目的成功；
- 所有人员都畅所欲言；
- 促进用户与开发团队之间达成共识；
- 能够揭露和解决那些妨碍项目成功的行政问题；
- 最终很快产生初步的系统定义。
- 专题讨论会准备
  - 参加会议人员：主持人、用户、技术人员、项目组人员
  - 安排日程
  - 通常在具有相应支持设备的专用房间进行
- 举行会议
  - 可能出现人员之间的责备或冲突，主持人应掌握讨论气氛并控制会场
  - 最重要的部分是自由讨论阶段，这种技术非常符合专题讨论会的气氛，并且营造一种创造性的和积极的氛围，同时可以获得所有相关者的意见
  - 分配会议时间，记录所有言论

7.2.3 现场观察/体验

- 用户可能无法有效全面的表达自己的需求，通过面谈和会议也难以获得完整信息；
- 在这种情况下，现场观察用户的工作流程有助于更深入全面了解需求
- 两种方式：
  - 被动观察：用户实地工作，需求分析人员在旁边看
  - 主动体验：需求分析人员直接参与用户的实际工作

7.2.4 头脑风暴

- 一般以8-12人最佳：
  - 人数太少不利于交流信息和激发思维；人数太多则不容易掌握，并且每个人发言的机会相对减少
- 明确分工：1名主持人、2名记录员
- 成功要点：
  - 自由畅谈
  - 延迟批判、禁止批评
  - 禁止批评、自我批评、自谦
  - 追求数量
- 会后：修剪、分组、排序
- 适用场合：产品型系统，需要具有创新性特征，尚未投放市场，无明确的客户。

7.3 需求获取的主要内容

对客户输入进行分类

- 业务需求：描述客户可以从产品中得到的资金、市场或其它业务利润的需求
  - 市场股票价格上升x%
  - 每年节约$Y
  - 替代了维护费用高的老一代系统Z
- 业务规则：一些活动只能在特定的条件下，由一些特定的人来完成时，该用户可能在描述一个业务规则
  - 如果一个药剂师在危险化学制品培训方面是可靠的，那么他就可以在一级危险药品清单上订购化学制品；
  - 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；
- 功能需求：
  - 客户所说的诸如“用户应该能<执行某些功能>”或者“系统应该<具备某些行为>”，这是最可能的功能需求。
  - 功能需求描述了系统所展示的可观察的行为，并且大多数是处于执行者—系统响应顺序的环境中。
  - 功能需求定义了系统应该做什么，它们组成了软件需求规格说明的一部分。
- 非功能需求：
  - 对系统如何能很好地执行某些行为或让用户采取某一措施的陈述就是质量属性，这是一种非功能需求。

- 外部接口需求：这类需求描述了系统与外部的联系。
  - 软件需求规格说明必须包括用户接口和通信机制、硬件和其它软件系统需求部分。
  - 客户描述外部接口需求包括如下习惯用语：
    - “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>”
- 约束条件
  - 是指一些合理限制设计者和程序员选择的条件，它们代表了另一种类型的非功能需求，必须把这些需求写入软件需求规格说明。
  - 尽量防止客户施加不必要的限制，但是一定的限制有助于提高产品质量属性。
  - 下面是客户描述限制的一些习惯用语：
    - “必须使用<一个特定的数据库产品或语言>”
    - “不能申请多于<一定数量的内存>”
    - “操作必须与<其它系统>相同”
    - “必须与<其它应用程序>一致”
- 数据定义：当客户描述一个数据项或一个复杂的业务数据结构的格式、允许值或缺省值时，他们正在进行数据定义。
  - 例如：“邮政编码由5个数字组成，后跟一个可选的短划线或一个可选的四位数字，缺省为0000”
  - 收集用户当前正在使用的各类数据表格、单据等，基于它们来获得对数据的详细定义。

8 撰写需求文档

软件需求规格说明(SoftwareRequirementsSpecification,SRS)

- 是具有一定法律效力的合同文档
- 清楚地描述软件在什么情况下，需要做什么，以及不能做什么
- 以输入/输出、输入到输出之间的转换方式来描述功能性需求
- 描述经过干系人磋商达成共识的非功能性需求
- 一般参考需求定义模板，覆盖标准模板中定义的所有条目
- 作为后续的软件评估依据和变更的基准

需求文档的组织形式

- 文档需要有逻辑组织结构
  - 例如：参照IEEE的模板
- 典型的组织形式包括

    - 按系统能够响应的各种外部环境情况来组织
    - 按系统特征来组织
    - 按系统的响应方式来组织
    - 按所管理的外部数据对象来组织
    - 按用户类型来组织
    - 按软件的工作模式来组织
    - 按子系统的划分来组织

软件需求规格说明SRS的风格

- 描述性的自然语言文本
    - 用户故事
- 从用例模型产生
    - 用例模型与需求转化可看成可逆的过程
    - 如果需求模型以用例的形式表示，我们可以逆向生成需求的完整集合
- 从需求数据库中生成
    - 商业需求数据库有内置的功能来生成经过筛选的需求规格说明
    - 从产品线需求规格数据库中生成特定产品的需求规格说明从用例模型产生
- 从混合模型中生成
    - 特征模型和用例模型

选择合适的需求规格说明方式

- 考虑以下两个具体项目场景:
    - 小型项目，1名程序猿，2个月的开发周期
        - 程序猿直接和用户对话，写了两页纸的备忘录
    - 大型项目，50名程序猿，2年的开发周期
        - 专门的团队进行需求建模与分析，撰写了500页的需求规格说明
-

	项目 A	项目 B
需求撰写的目的	凝练工程师对问题的理解；获取用户反馈	文档本身就是交付物；为研发人员提供丰富细节
管理方面的用途	资源的分配与规格说明书无关；相关资源已经就位	基于规格说明进行估算和任务进行规划
预期读者	主用户：作者本人；从用户:客户	主用户：程序员，测试人员，项目经理；从用户：客户

生成不同风格SRS的方法总览

方法	使用时机	适合	效果	需要的资源
描述性文本	小项目	小产品的非正式说明	中	有良好写作能力的分析师
从用例模型生成	大中型项目	产品线的正式规格描述	中	有建模能力，好的过程标准，建模工具的分析师
从需求数据库生成	大中型项目	产品线的正式规格描述	小	需求数据库，有数据库管理能力的分析师
从混合模型生成	中小型项目	单个产品的定义	中小	有良好协作技巧，需求工程，数据库管理，建模能力的分析师

用户手册作为SRS

- 撰写用户手册作为一种性价比高的一箭双雕的方法，同时获得SRS和用户手册
- 用户手册作为SRS对于和用户交互的系统是比较有效的，这样系统由交互驱动
- 好的手册描述了所有用例的所有场景
  - FredBrooks的《人月神话》中描述了将用户手册作为需求规格说明书的做法
  - 据说在苹果的第一台电脑的编码工作开始之前用户手册就写好了
  - 但是...用户手册并没有描述
- 非功能性需求
- 不和用户交互的功能性需求
  - 比如：函数计算、过滤器或者翻译工具的时候

用户手册大纲

- 介绍
  - 产品总览及基本原理
  - 术语和基本特征
  - 展示格式与报表格式的总结
  - 手册的大纲
- 开始
  - 开始指令
  - 帮助模式
  - 样例运行
- 操作模式:
  - 命令行/对话框/报告
- 高级特性
  - 命令语法和系统选项

需求规格说明的用户

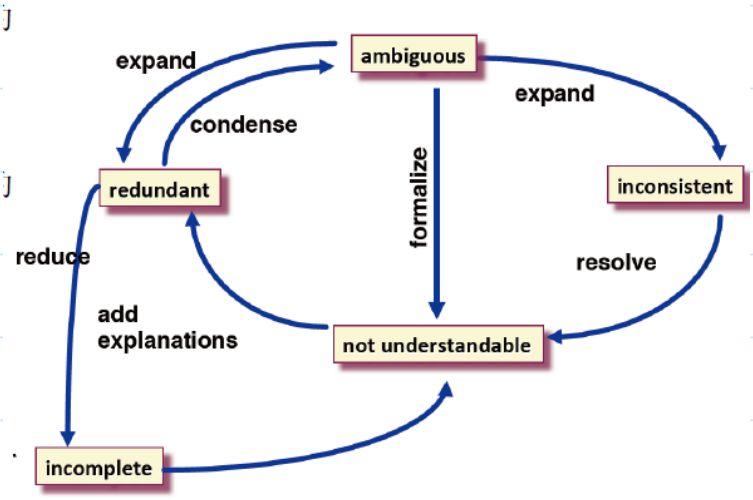
高质量需求规格说明

一个高质量的需求规格说明产品总览及基本原理

- 是所有需求的集合
- 描述产品要提供的所有功能
- 是软件系统解决方案的商业合同的基础
- 是测试计划的基础
- 定义产品需求的度量标准
- 是产品需求跟踪的先决条件
- 影响开发产品的项目计划

高质量需求规格说明的评价标准

- 正确性=经过验证的
- 无歧义
- 完整的
- 可测试=可以证明的
- 可修改的
- 可跟踪的
- 易理解
- 一致的
- 有序的
- 项目或产品特定的其他特征

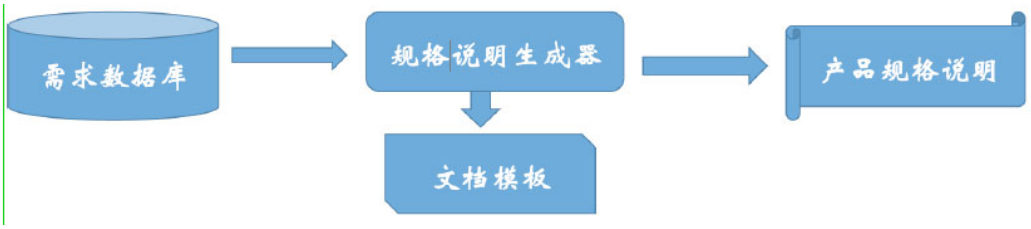


简洁

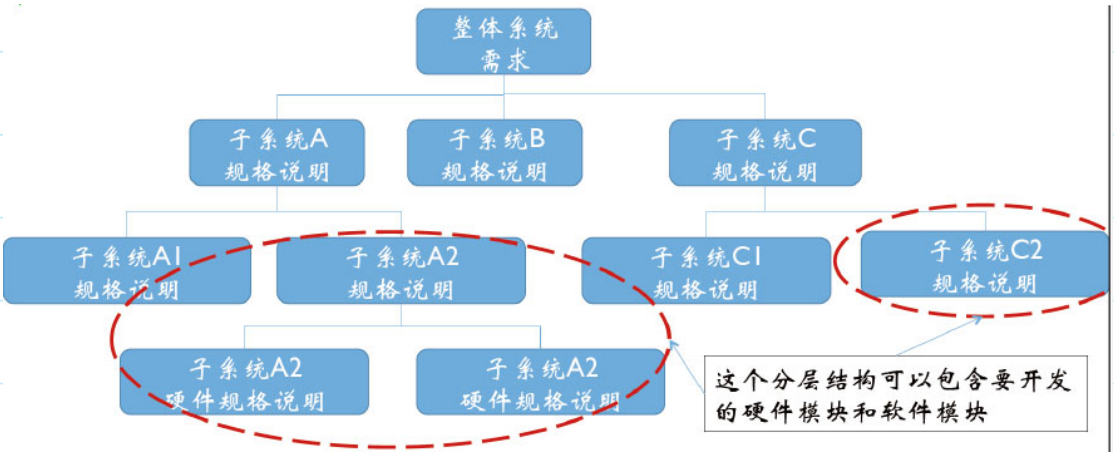
- 定义：需求描述尽可能简洁
  - 描述了系统的一个独立特征
  - 除了必需的信息外没有包含其他信息
  - 用清晰、简单、可理解的词表述
  - 避免“应该”、“可以”、“可能”之类的用词
- 举例:
  - “急救电话的响应应本着先到先响应的原则”相对“急救电话应按照其拨入的次序存入一个先入先出的等待队列当中，并且按照进入队列的次序逐一应答”是简洁的

需求规格说明生成过程

- 创建任何类型的需求规格说明最优先的方式是通过需求数据库生成它
- 不直接修改规格说明，修改数据库中的需求并且重新生成规格说明
-



需求规格说明的结构

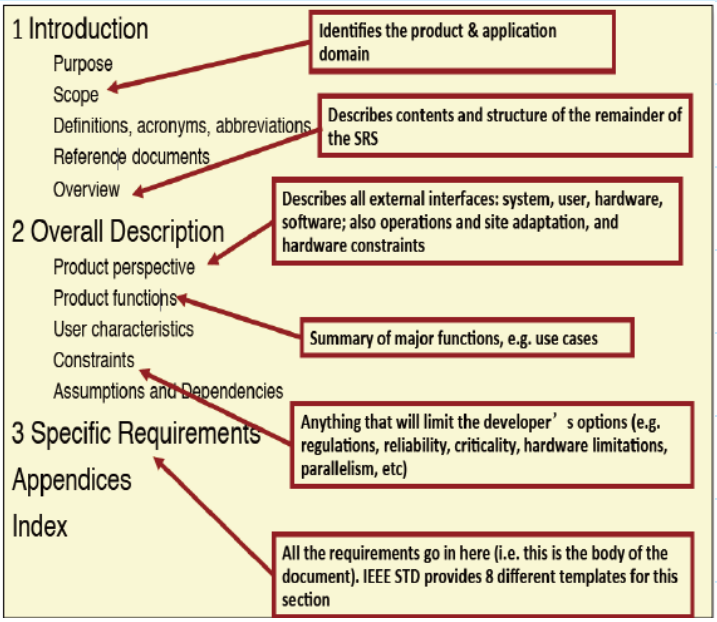
- 

需求规格说明SRS模板

- SRS需要根据预先定义的标准章节模式来组织，即根据模板来撰写
- SRS的模板使得撰写统一的SRS变得简单
- 对于QA人员来说定义SRS指标变得简单
- 模板也适用于业务需求和系统需求
- 模板可以被用于半自动地从需求数据库或者用例模型生成SRS

IEEE830SRS模板大纲

- 介绍
- 术语表
- 用户需求规格说明
- 系统结构
- 系统需求规格说明
- 系统模型
- 系统的演化
- 附录
- 索引

- 

IEEE830SRS模板第三部分

- 3.1 外部接口
  - 3.1.1 用户接口
  - 3.1.2 硬件接口
  - 3.1.3 软件接口
  - 3.1.4 通信接口
- 3.2 功能需求描述
  - *按系统的操作模式、用户分类、特征分类来定义:*
  - 3.2.1 模式 1
    - *3.2.1.1 功能需求 1.1*
    - ...
  - 3.2.2 模式 2
    - *3.2.2.1 功能需求 2.1*
    - ...
  - ...
  - 3.2.n 模式 n
- 3.3 性能需求
  - *要用可度量的方式来表达!*
- 3.4 设计约束
  - *3.4.1 标准依从性*
  - *3.4.2 硬件限制*
  - *etc.*
- 3.5 软件质量属性
  - 3.5.1 可靠性
  - 3.5.2 可用性
  - 3.5.3 安全性
  - 3.5.4 可维护性
  - 3.5.5 可移植性
- 3.6 其他需求

SRS模板的优缺点

- 优点
  - 模板提高效率
  - 在有模板的情况下，面对一个完整的大纲，不容易遗漏重要的信息
- 缺点
  - 并非对于所有的系统，模板的章节设计都是类似的
  - 如果仅仅为了满足标准，而填写模板的所有章节，在不相关的章节，会加入一些没有意义的内容
  - 读者很难将这些无意义的文字和真正的需求分开

SRS小结

- 尽快开始写需求
- 确定哪些属性将被用于分类和细化需求
- 产生一个初始版本来刺激反馈
- 询问用户往往比咨询专家更有用
- 撰写需求时需要遵循的法则：
    - 使用简单、直接的语言
    - 撰写可测试的需求
    - 使用定义好的并达成共识的术语
    - 一次只写一项需求