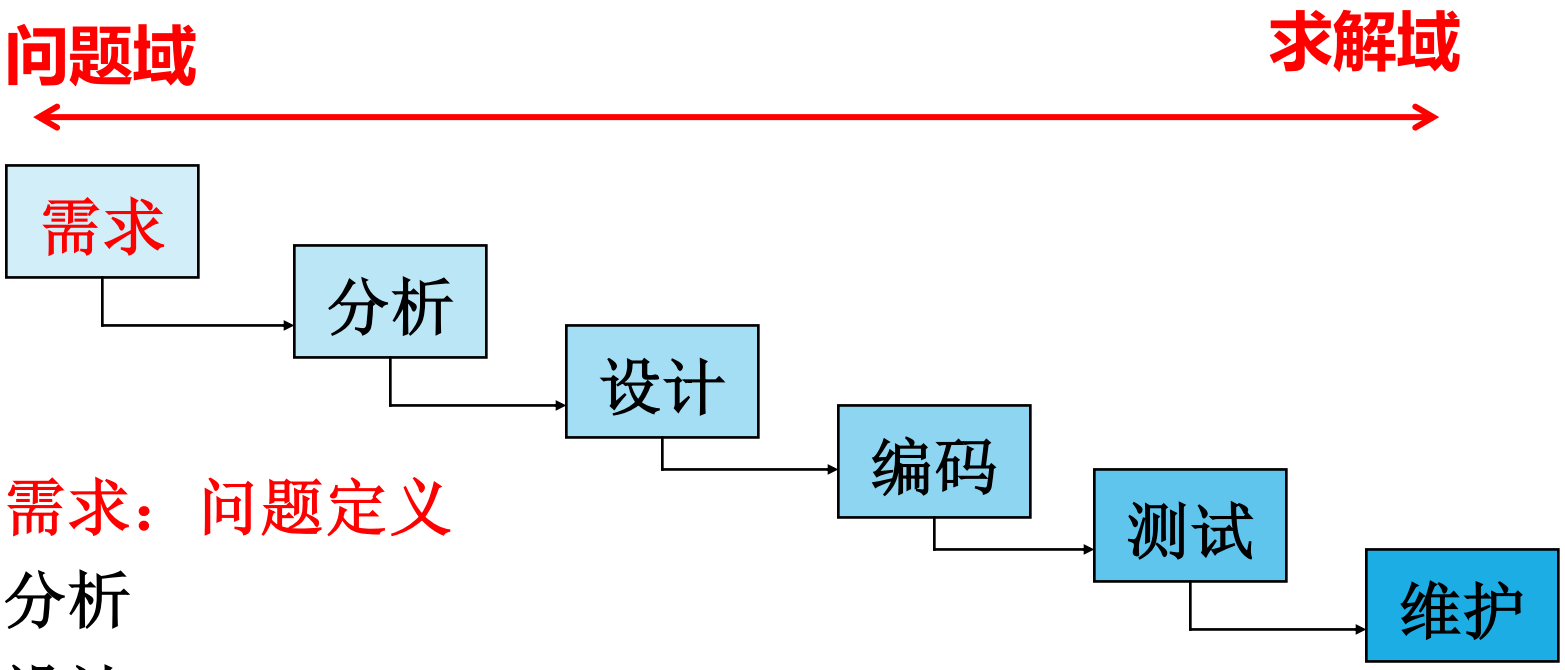
南京大学
NANJING UNIVERSITY

软件需求与需求工程

张天
SEG - Software Engineering Group
2021年秋季

需求的位置



- 需求：问题定义
- 分析
- 设计
- 编码
- 测试
- 维护

需求工作流的目标：
确定客户的需要，即目标系统应有什么功能。

提纲

- 软件需求
- 需求工程过程
  - 需求获取
  - 需求分析
  - 需求规约与文档化
  - 需求验证：需求评审
  - 需求管理技术与工具

穿越隐喻

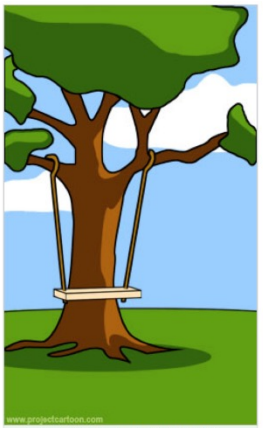
- 以**穿越**为隐喻来理解将计算机作为工具的问题
  - 计算机属于工科性的学科，相关研究是为了更好地将它作为工具应用到各种**领域**中
  - 将计算机作为工具必须要解决软件开发的问题
    - 软件是计算机的灵魂！
  - 必然涉及到两个领域：
    - 问题域：应用计算机的领域
    - 求解域：计算机领域
  - 软件开发就是要完成在这两个领域之间的穿越

穿越隐喻

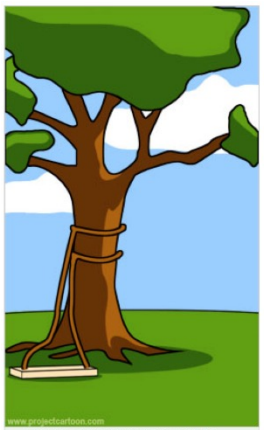
- 穿越的宽度就代表了软件的复杂性
  - 问题越复杂，则需要越多的中间层（就像驿站）
- 在计算机领域有一句著名的话：
  - *“在计算机领域，任何一个复杂问题都可以通过增加一个抽象层来解决！”*

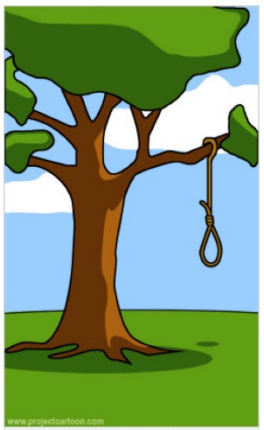
经典的需求秋千图


你的“上帝”是怎么期望的。

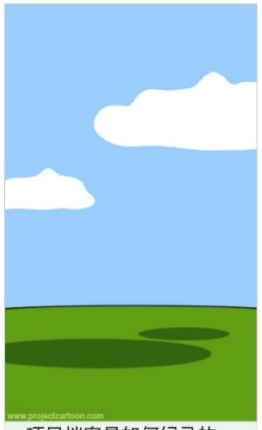

项目经理是如何理解的。

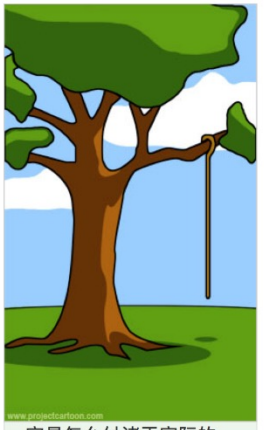

设计师么是怎么设计的。

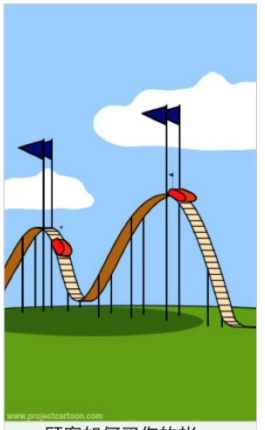

程序员们是如何开发的。

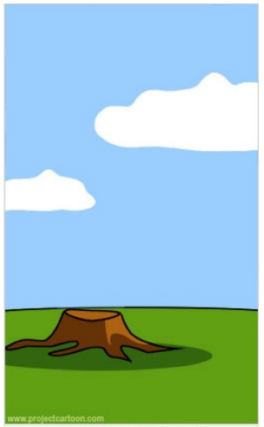

测试员们得到的。


你的商业顾问是怎么形容的。


项目档案是如何纪录的。


它是怎么付诸于实际的。


顾客如何买你的帐。


它是如何被支持的。

广告是如何做的。


客户到底需要的是什么。

秋千图的故事

- **客户**：我家有三个小孩，我须要一个能三个人用的秋千。它是由一绳子吊在我园子里的树上。
- **项目经理**：秋千这东西太简单了，秋千就是一块板子，两边用绳子吊起来，挂在树上的两个枝子上。
- **分析员**：这个无知的项目经理，两个树枝上挂上秋千哪还能荡漾起来吗？除非是把树从中截断再支起来，这样就满足要求了。
- **程序员**：两条绳，一块板，一棵大树，接在树的中段，太简单了，搞定！
- **商业顾问**：您的需求我们以完成，我们通过人体工学，工程力学多方面研究。本着为顾客服务出发，我们的秋千产品在使用时给你如同游乐园里的过山车一样刺激，如同你在地面上坐沙发一样舒适与安全！
- **文档管理员**：这么小的工程没有文档很正常，只要需求说明书与合同就可以了。
- **实施人员**：我们的产品用户自己都可以完成安装，只要把绳子系在树上就可以了。

软件需求

- 软件需求是软件系统应该提供的服务，或者对系统的约束的一个高层抽象描述
  - 待开发软件产品的目标用户对该软件产品的**功能**、性能、设计约束和**其它**方面的期望和要求
- IEEE软件工程标准词汇表（1997年）中定义需求为：
  - 用户解决问题或达到目标所需的条件或权能（Capability）。
  - 系统或系统部件要满足合同、标准、规范或其它正式规定文档所需具有的条件或权能。
  - 一种反映上面（1）或（2）所描述的条件或权能的文档说明。
- 需求层次
  - 领域需求（业务需求）：应用领域的要求
  - 用户需求：表达高层的概要需求
    - 从用户角度描述对系统的需求，用户使用系统必须完成的任务
  - （软件）系统需求：描述系统应该提供的服务及其约束
    - 功能需求：系统提供的服务、系统的行为
    - 非功能需求：对系统提供的服务及行为的约束

常见的软件开发问题

- 不能满足用户或业务需要
- 需求混乱
- 模块集成困难
- 系统难以维护
- 缺陷发现太迟
- 低质量、难使用
- 性能问题
- 小组人力部署不协调
- 提交问题

根本原因分析

- 需求不正确
- 交流不明确
- 架构不合理
- 软件太复杂
- 未检不一致
- 测试不充分
- 评估太主观
- 方法太原始
- 变更未控制
- 不够自动化

解决办法

- 迭代式开发
- 管理需求
- 基于组件的架构
- 可视化建模
- 连续验证质量
- 管理变更

解决问题的思路

问题

- 不能满足用户或业务需要
- 需求混乱
- 模块集成困难
- 系统难以维护
- 缺陷发现太迟
- 低质量、难使用
- 性能问题
- 小组人力部署不协调
- 提交问题

根本原因

- 需求不正确
- 交流不明确
- 架构不合理
- 软件太复杂
- 未检不一致
- 测试不充分
- 评估太主观
- 方法太原始
- 变更未控制
- 不够自动化

解决办法

- 迭代式开发
- 管理需求
- 基于组件的架构
- 可视化建模
- 连续验证质量
- 管理变更

讨论

- 软件需求关注用户的期望、要求和需要，不是解决方案
- 并不是所有方面的要求都是软件需求
  - 功能、性能、设计约束、时间进度等
  - 例如，重量、软件大小等不是用户需求
- 并不是所有用户的期望和要求都是软件需求
  - 用户需求必须中肯，有意义、可实现

需求的重要性

Frederick Brooks在他1987年经典文章“No Silver Bullet”中阐述了需求的重要性：

- 开发软件系统最困难的部分就是准确说明开发什么。最困难的概念性工作是编写出详细的需求，包括所有面向用户、面向机器和其它软件系统的接口。此工作一旦做错，将会给系统带来极大的损害，并且以后对它修改也极为困难。

- 软件开发的基础和前提
  - 只有在明确了软件需求之后才能开展有针对性的软件开发工作
  - 没有需求无法进行设计和编码
- 制定软件开发计划的基础
  - 只有知道你想做什么，才能知道做这些东西需要多少工作量？
  - 不知道软件需求也就不知道工作量的大小，因而不能制定计划
- 最终目标软件系统验收的标准
  - 只有知道你想做什么，才能知道你最终是否做好了
  - 没有定义明确的需求，就不知道最终基于什么进行验收

获取需求的复杂性

- 系统复杂和庞大
  - 如何将软件需求得到？描述清楚？
- 片面, 不完全
  - 如何保证得到了所有的软件需求？
- 模糊, 不准确
  - 如何保证把需求说清楚和准确？
- 不一致, 歧义
  - 如何保证所描述的需求是不矛盾的？
- 及时性
  - 当需求变更时，如何让相关人员都知道需求已经变更？
- 软件需求变动带来的问题
  - 波动性
  - 放大性

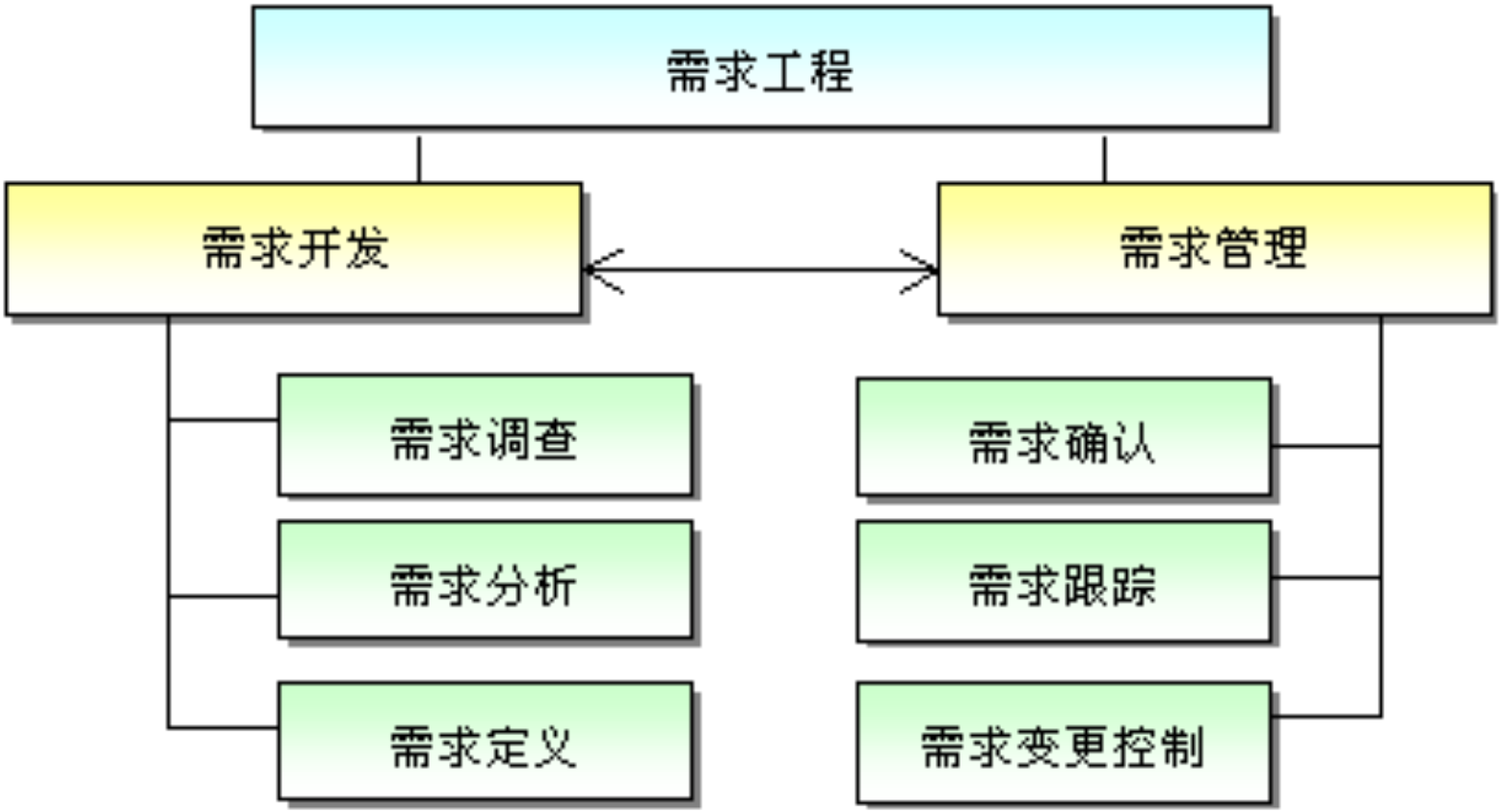
需求的发展趋势

- 技术层面
  - 需求分析方法、技术和工具
    - 方法：数据流、结构化方法、面向对象方法
    - 技术：抽象、建模、多视点、原型、……
    - 工具：UML，Rose，Word，Excel，RequisitePro， RRC
- 管理层面
  - 对需求分析中的人、活动和产品进行管理
- 形成新的研究领域： 需求工程

需求工程过程

- 软件工程子领域——需求工程(requirement engineering, RE)
- 目标：创建和维护系统的需求文档
  - 确切地定义用户要求解决的问题，也就是确定问题的性质、工程的目标和规模
- 需求工程过程
  - 可行性研究
  - 需求获取与分析
  - 需求描述
  - 需求验证
  - 需求管理

需求工程活动组织框架



可行性分析

- 经济可行性
- 技术可行性
- 法律可行性
- 不同的方案

需求的获取与分析

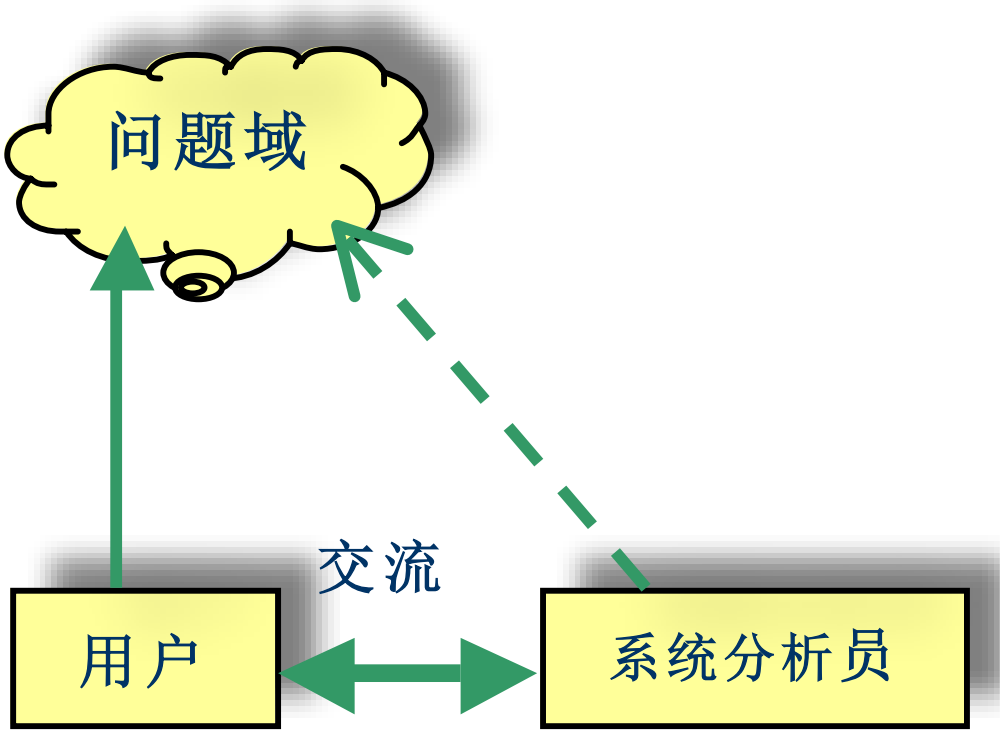
- 需求分析
  - 是指从用户处获得需求、形成与用户需求相一致的、可供阅读的软件需求规格说明书的过程
- 需求分析的任务
  - 通过对应用问题及其环境的理解和分析，准确、一致和完全地刻划用户需求，并达成一致

获取软件需求

- 任务
  - 从用户处收集、获取软件需求, 帮助用户发现潜在的软件需求
- 来源
  - 软件用户 / 客户
- 成果
  - 需求描述

获取软件需求

- **技术手段**
  - **访谈**
  - **会议**
  - **参观**
  - **实践**
  - **……**



客户 / 用户权利

- 要求分析人员使用符合客户语言习惯的表达。
- 要求分析人员了解客户系统的业务及目标。
- 要求分析人员组织需求获取期间所介绍的信息，并编写软件需求规格说明。
- 要求开发人员对需求过程中所产生的工作结果进行解释说明。
- 要求开发人员在整个交流过程中保持和维护一种合作的职业态度。
- 要求开发人员对产品的实现及需求都要提供建议，拿出主意。
- 描述产品使其具有易用、好用的特性。
- 可以调整需求，允许重用已有的软件组件。
- 当需要对需求进行变更时，对成本、影响、得失（trade-off）有个真实可信的评估。
- 获得满足客户功能和质量要求的系统，并且这些要求是得到开发人员同意的

客户 / 用户义务

- 给分析人员讲解业务及说明业务方面的术语等专业问题。
- 抽出时间清楚地说明需求并不断完善。
- 当说明系统需求时，力求准确详细。
- 需要时要及时对需求做出决策。
- 要尊重开发人员的成本估算和对需求的可行性分析。
- 对单项需求、系统特性或使用实例划分优先级。
- 评审需求文档和原型。
- 一旦知道要对项目需求进行变更，要马上与开发人员联系。
- 在要求需求变更时，应遵照开发组织确定的工作过程来处理。
- 尊重需求工程中开发人员采用的流程（过程）。

需求分析

- 对收集的用户软件需求，进行建模、分析，发现并纠正不一致、不准确和不全面的软件需求，形成准确的需求描述与需求模型
- 需求分析与建模方法
  - 结构化分析方法
    - 数据流模型
  - 面向对象分析方法
    - 面向对象模型
  - 原型方法
  - 其他。。。

需求分析的原则

- 必须要理解和描述问题域
- 必须确定软件所要实现的功能
- 必须描述软件应具有的行为
- 必须明确软件实现功能时需要满足的约束（非功能属性）
- 必须要以一种分层的方式来对软件需求进行分析和建模

需求规约

- 根据软件需求描述和软件需求模型，撰写软件需求规约（规格说明书）
  - 功能与行为需求
  - 非功能需求
  - 设计约束（如硬件、软件、网络等环境）
  - 其他（如开发周期）
- 需求规约编制规范（模板）
  - IEEE／ANSI 830－1998
  - GB／T 8586－2008

描述语言

- 需求规约描述语言
  - 自然语言
  - 伪代码语言
  - 形式语言—形式规约
    - 需求描述：形式化语言，Z，CSP，T / I / H Automata
    - 需求模型：半形式化语言，UML

软件需求规约

1. 概述
2. 信息描述
(1) 数据流程图
(2) 数据字典
(3) 数据结构
(4) 系统接口说明
(5) 内部接口

3. 功能说明
(1) 功能
(2) 处理说明
(3) 设计的限制
4. 检验标准
(1) 性能界限
(2) 测试种类
(3) 预期的软件响应
(4) 应考虑的特殊问题

5. 参考文献
6. 附录

需求验证

- **目标：**确保需求规约（描述 / 模型）满足下列质量属性
  - **正确性、准确性**
  - 有效性、现实性
    - 是否合理、可实现
  - **无歧义性**
  - **一致性**
  - **完全性**（全面性、完整性）：没有遗漏
  - **可验证性**
  - **可理解和可修改性**
  - **可追踪性**
  - 认同
    - 共同、相互认可
    - 符合为过程、项目和产品建立的标准

需求验证

- 手段：
  - 评审
    - 评审检查单
    - 评审组织：工程师、客户、用户、其他相关人员
  - 自动验证
    - 形式化需求规格说明
    - 形式化属性

需求评审检查单

- 所规定的软件目标和任务与系统的目标和任务相符合吗？
- 与所有系统成分的重要接口都已被描述了吗？
- 研制项目的数据流程图、数据字典、数据结构充分确定了吗？
- 图表都清楚吗？每个图表在不加补充说明的情况下能被理解吗？
- 主要功能在规定的范围之内吗？每一种功能被充分说明了吗？

需求管理

- 需求管理
  - 帮助项目组在项目进展中标识、控制和跟踪需求以及变更需求的一组活动。
- 需求的配置管理
  - 标识需求
  - 需求追踪
  - 变更控制
- 辅助工具
  - RRC（Rational Requirement Composer）

需求跟踪

- 需求跟踪的目的是建立与维护“需求－设计－编程－测试－维护”之间的一致性，确保所有的工作成果符合用户需求。
- 需求跟踪有两种方式：
  - 正向跟踪。检查软件需求规约中的每个需求是否都能在后继工作成果中找到对应点。
  - 逆向跟踪。检查设计文档、代码、测试用例等工作成果是否都能在软件需求规约中找到出处。
- 正向跟踪和逆向跟踪合称为“双向跟踪”。不论采用何种跟踪方式，都要建立与维护需求跟踪矩阵（即表格）。需求跟踪矩阵保存了需求与后继工作成果的对应关系。

需求变更

- 需求变更控制
  - 变更目的： 产品更加符合用户的需求
  - 需求发生变更的起因主要有：
    - 原先的需求不足或错误
    - 市场发生了变化
  - 问题：对项目开发小组而言，变更需求意味着要调整资源、重新分配任务、修改前期工作成果等，开发小组要为此付出较重的代价。如果每次需求变更请求都被采纳的话，这个项目也许永远不能按时完成。
  - 解决办法：对变更进行控制
    - 原则：建立需求变更控制规则

讨论

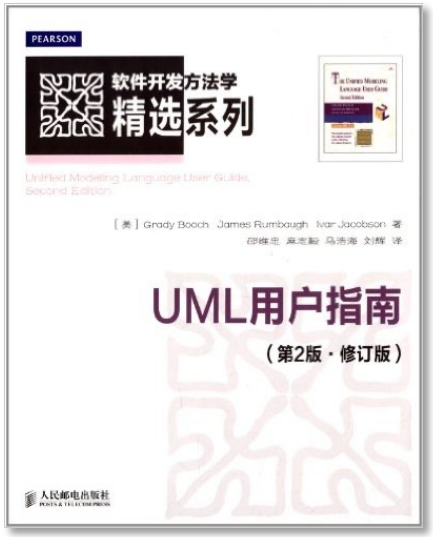
- 软件需求分析是软件生存期的一个重要阶段，是软件开发项目得以成功的基础。其最根本的任务是确定为了满足用户的需要软件系统必须做什么。
- 软件需求分析是一个不断发现和决定的过程，在此过程中，软件开发者和用户同样起着重要的作用。
- 在需求分析与说明过程中，需要大量交换意见，其间充满着传错信息和发生误解的可能性：
- “我知道你相信你明白了你认为我所说的是什么，但是我不能肯定你是否意识到你听到的并不是我所指的意思……”。

讨论

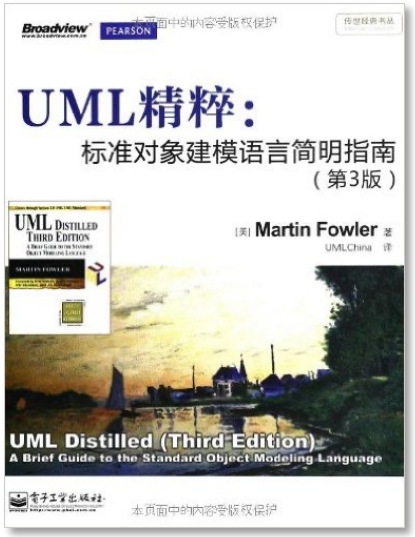
- 系统分析员
  - 善于领会一些抽象的概念，重新整理使之成为各种逻辑成分，并根据各种逻辑成分综合出问题的解决办法；
  - 善于从各种相互冲突或混淆的原始资料中吸取恰当的论据；
  - 能够理解用户的环境及领域知识；
  - 具备把系统的硬件和软件部分应用于用户环境的能力；
  - 具备良好的书面和口头形式进行讨论和交换意见的能力；
  - 具有“既能看到树木，又能看到森林”的能力。
  - ……

推荐书籍

软件建模与设计:UML、用例、模式和软件体系结构



UML用户指南(第2版)(修订版)



UML精粹:标准对象建模语言简明指南(第3版)