book17-需求管理

1. 需求管理

1. 需求基线应该成为后续软件系统开发的工作基础和粘合剂
    1. 项目管理者根据需求安排、监控和管理项目计划。
    2. 开发者依据需求开发相应的产品功能和特性。
    3. 测试人员按照需求执行系统测试和验收测试。
    4. 客户和顾客依照需求验收最终产品。
    5. 维护人员参考需求执行产品的演化。

1.1. 意图

1. 需求的**影响力**：整个后续的产品生命周期 VS 需求开发阶段
2. 需求规格说明文档：后续的开发工作都应该以软件需求规格说明文档的内容为标准和目标来进行
3. **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

1.2. 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

1.3. 任务

1. 交流涉众需要什么
2. 将需求应用、实施到解决方案
3. 驱动设计和实现工作
4. 控制变更

5. 将需求分配到子系统
6. 测试和验证最终产品
7. 控制迭代式开发中的变化
8. 辅助项目管理

1.4. 活动

1. 需求管理的3个活动：维护需求基线、实现需求跟踪和控制变更。

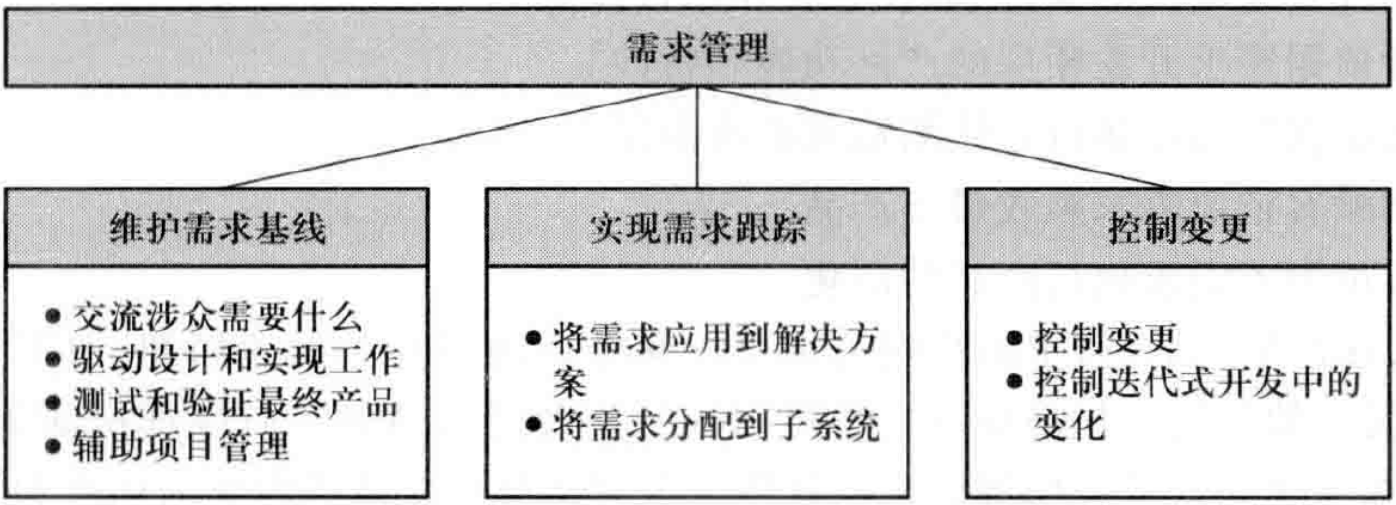


图 17-1　需求管理活动

2. 维护需求基线

2.1. 需求基线

1. 基线：已经通过正式评审和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制过程才能修改它
2. 基线是被**明确和固定**下来的需求集合，是项目团队需要在某一特定产品版本中实现的**特征和需求集合**
3. 需求基线是需求开发过程的成果总结，在后续产品生命周期中持续发挥作用。
4. 一般将需求基线编写成文档纳入配置管理。

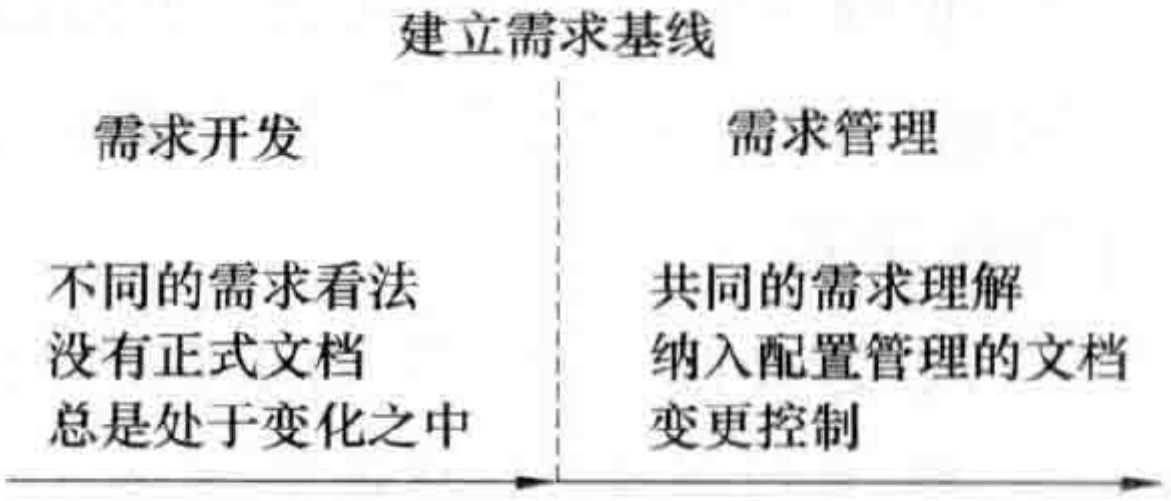


图 17-2　需求基线示意

2.2. 需求基线的内容

1. 标识符(ID)，为后续的项目工作提供一个共同的交流参照。
2. 当前版本号(Version)，保证项目的各项工作都建立在最新的一致需求基础之上。
3. 源头(Source)，在需要进一步深入理解或者改变需求时，可以回溯到需求的源头。
4. 理由(Rational)，提供需求产生的背景知识。
5. 优先级(Priority)，后续的项目工作可以参照优先级进行安排和调度。
6. 状态(Status)，交流和具体需求相关的项目工作状况。
7. 成本、工作量、风险、可变性(Cost、Effort、Risk、Volatility)，为需求的设计和实现提供参考信息，驱动设计和实现工作。
8. 需求创建的日期
9. 和需求相关的项目工作人员，包括需求的作者、设计者、实现者、测试者等
10. 需求涉及的子系统
11. 需求涉及的产品版本号
12. 需求的验收和验证标准
13. …

2.3. 需求基线的维护

2.3.1. 配置管理

1. 标识配置项
    1. **递增数值**，例如1，2，…x；

    2. **层次式数值编码**，例如1.1.1，1.2.1，…x.y.z；
    3. **层次式命名编码**，例如Order.Place.Date，Order.Place.Register，…Task.Step.Substep
2. 版本控制
    1. 每一条单独的需求需要进行版本控制
    2. 相关的需求文档也需要进行版本控制
    3. 每一个刚纳入配置管理的软件需求项赋予一个初始的版本号，并在需求发生变更时更新需求的版本号。
3. 变更控制
4. 访问审计：应当易于被项目涉众访问，记录和审计访问的情况，不应当很随意访问需求基线。
5. 状态报告：反映需求基线的成熟度(变化的幅度越大，成熟度越低)、稳定性(改变的次数越多，稳定性越差)等

2.3.2. 状态维护

表 17-1　需求的状态类别

状态	定义
已提议(proposed)	该需求已被有相应权限的人提出
已批准(approved)	该需求已经被分析，它对项目的影响已进行了估计，并且已经被分配到某一特定版本的基线中。关键涉众已同意包含这一需求，软件开发团队已承诺实现这一需求
已实现(implemented)	实现这一需求的系统组件已经完成了设计和实现。这一需求已经被跟踪到相关的设计元素和实现元素

状态	定义
已验证(verified)	已在集成产品中确认了这一需求的功能实现是正确的。这一需求已经被跟踪到相关的测试用例。这一需求目前可以被认为是已完成了
已删除(deleted)	已批准的需求又从需求基线中取消了。要解释清楚为什么要删除这一需求，以及是谁决定删除的
已否决(rejected)	需求已被提议，但并不在下一版本中实现它。要解释清楚为什么要否决这一需求，以及是谁决定否决的

3. 实现需求跟踪

1. 避免在开发过程或者演化过程中与需求基线不一致或者偏离的风险
2. 需求跟踪是以软件需求规格说明文档作为基线，在向前和向后两个方向上，描述需求以及跟踪需求变化的能力。

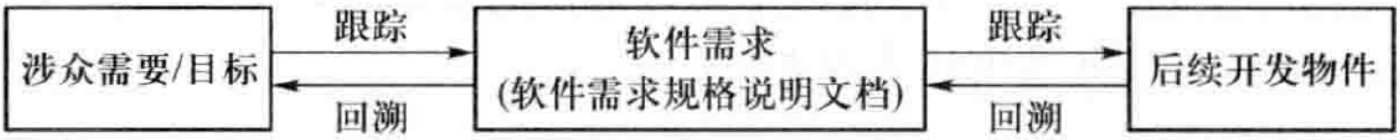


图 17-3　需求跟踪的联系类型

3.1. 需求跟踪

3.1.1. 前向跟踪

1. 前向跟踪是指被定义到软件需求规格说明文档之前的需求演化过程
    1. 向前跟踪到需求：**说明涉众的需要和目标产生了哪些软件需求**
    2. 从需求向后回溯：**说明软件需求来源于哪些涉众的需要和目标**

3.1.2. 后向跟踪

1. 后向跟踪是指被定义到软件需求规格说明文档之后的需求演化过程
    1. 从需求向前跟踪：说明软件需求是如何被**后续的开发物件支持和实现**的
    2. 回溯到需求的跟踪：说明各种系统开发的物件是**因为什么原因(软件需求)而被开发出来的**

3.2. 需求跟踪的用途

1. 需求的后向跟踪可以帮助项目管理者
    1. 评估需求**变更的影响**。
    2. 尽早发现需求之间的**冲突**，避免未预料的产品延期。
    3. 可以收集**没有被实现**的需求，并估算这些需求需要的工作量。
    4. 发现可以复用的**已有组件**，从而降低新系统开发的时间和精力。
    5. 明确需求的**实现进度**，跟踪项目的状态。
2. 需求的后向跟踪可以帮助客户和用户：
    1. 评价针对用户需求的**产品的质量**。
    2. 可以确认成本上没有(昂贵的)**镀金浪费**。
    3. 确认验收测试的**有效性**。
    4. 确信开发者的**关注点**始终保持在需求的实现上。
3. 需求跟踪中针对具体需求的设计方案选择、设计假设条件以及设计结果等信息可以帮助设计人员
    1. 验证设计方案正确的满足了需求。
    2. 评估需求变更对设计的影响。
    3. 在设计完，成很久之后仍然可以理解设计的原始思路。
    4. 评估技术变化带来的影响。

    5. 实现系统组件的复用。
4. 需求跟踪信息还可以帮助维护人员
    1. 评估某一个需求变化时对其他需求的影响
    2. 评估需求变化时对实现的影响。
    3. 评估未变化需求对实现变更的允许度。

3.3. 需求跟踪的内容

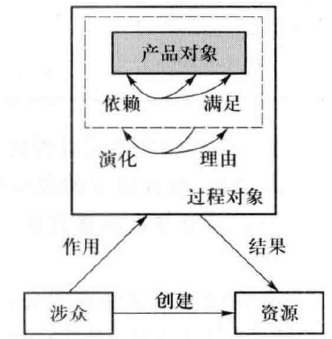


图 17-4　需求跟踪策略的 3 个层次

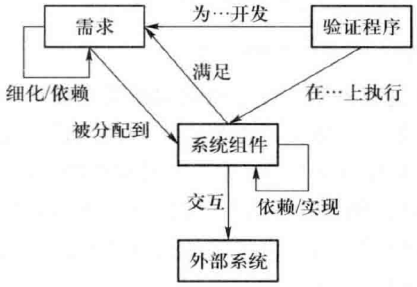


图 17-5　产品组件之间的跟踪关系示意

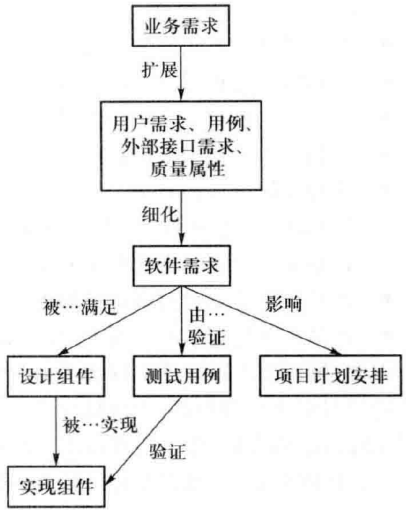
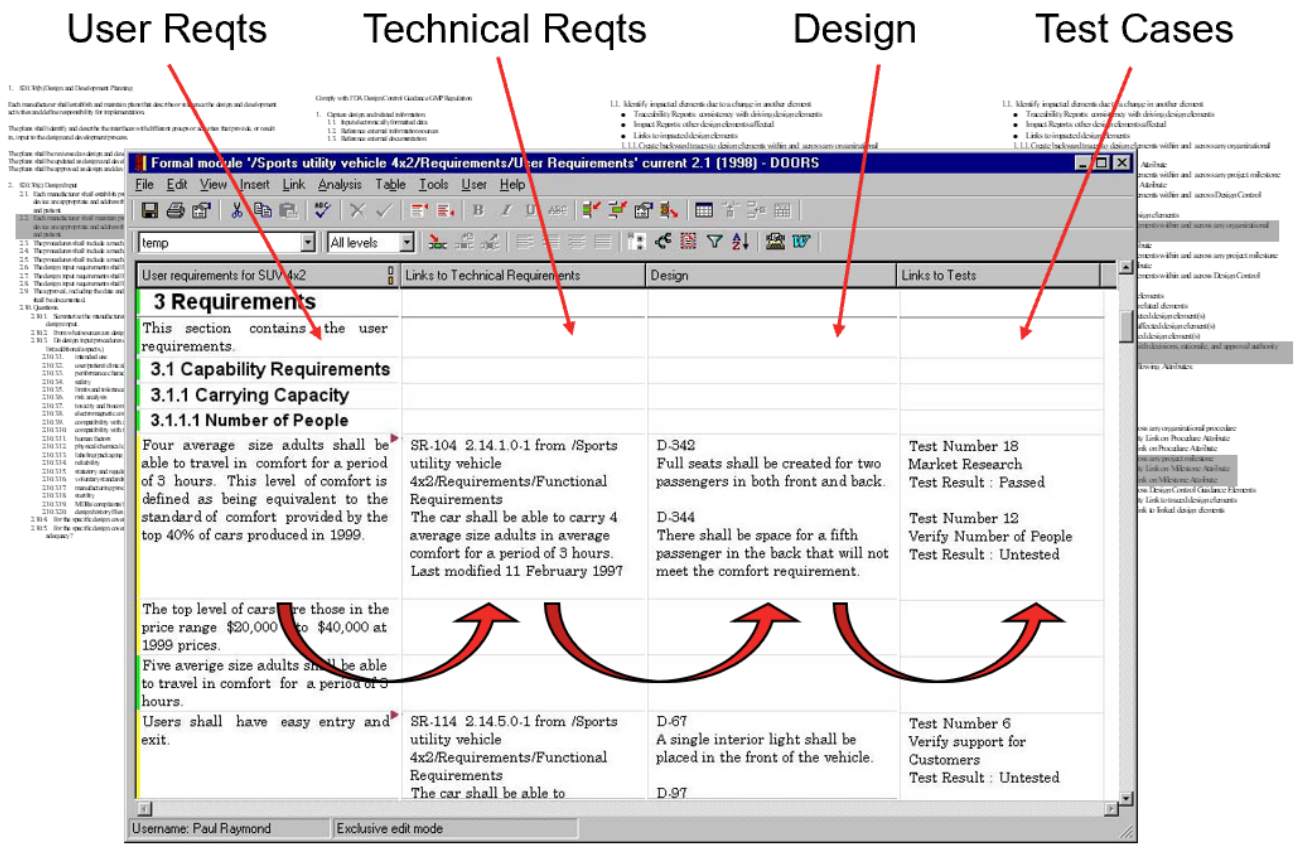


图 17-6　常见的产品组件之间的跟踪关系

1. 需求跟踪的内容依赖于项目的跟踪策略。
2. 最低层次上，需求跟踪仅仅是捕获了产品内部各个系统组件之间的依赖、满足和实现关系。
3. 最低层次上的需求跟踪策略使用者称为低端用户。
4. 实现了工作背景和过程背景捕获的需求跟踪实现者称为高端用户。
5. 产品本身所包含的各种联系、项目的组织过程

3.4. Traceability view



End-to-end visual validation in a single view

3.5. 实现方法

1. 需求跟踪的实现方法主要有矩阵、实体关系模型和交叉引用。
2. 需求跟踪矩阵是最常用的，如下图所示。
    1. 优点：跟踪信息清晰易懂
    2. 缺点：只能表达二元的跟踪关系。

表 17-2　需求跟踪矩阵示例

用户需求	功能性需求	设计组件	实现组件	测试用例
UC-28	Catalog.query.sort	Class catalog	Catalog.sort( )	Search.7 Search.8
UC-29	Catalog.query.import	Class catalog	Catalog.import( ) Catalog.validate( )	Search.12 Search.13 Search.14

3. 实体关系模型：使用实体关系模型来描述需求的跟踪联系，图17-5可以被认为是一个元模型示例。
    1. 优点：表达多元的跟踪关系，并且建立的跟踪信息可以利用关系数据库来实现、易于查询和维护。

    2. 缺点：不够直观，需要实体关系基础。
4. 交叉引用：文档之间建立跟踪关系。
    1. 优点：直接，利于使用
    2. 缺点：只适用于需求文档的处理

3.6. 建立过程

1. 认识到需求跟踪的重要性，明确需求跟踪需要解决的问题
2. 说明需求跟踪过程的目标
3. 明确需要捕获的跟踪联系
4. 组织提供资源支持和技术支持
5. 制定有效的过程策略：需求跟踪过程与实际的项目开发工作融合，作为项目开发工作的一部分。
6. 便利需求跟踪信息的使用：为客户、项目管理者以及开发者等项目涉众提供便利的使用途径。

3.7. 需求依赖

1. 大多数的需求并不是**完全独立**的，它们在一种**复杂**的机制中互相影响
2. 需求之间的依赖联系对很多项目开发工作都有重要影响。例如在表17-3所示的需求依赖关系示例当中，R1依赖于R3和R4，那么在实现、变更或者复用R1时，就必须将R3和R4也考虑在内。
3. 需求依赖联系的特殊性并不在于它的重要性，而在于它是难以发现、建立和维护的
4. 需求交互作用管理：用于**发现、管理和部署**(disposition)需求之间关键联系的活动

表 17-3　依赖联系的需求跟踪矩阵示例

依赖	R1	R2	R3	R4	R5	R6
R1			*	*		
R2					*	*
R3				*	*	
R4		*				
R5						*
R6						

4. 需求变更控制

4.1. 需求变化

1. 需求开发时一个获取、明确并定义需求的过程，并且需求并不是在需求开发结束之后就会固定不变的。
2. 需求的变化是正当和不可避免
    1. 问题发生了改变
    2. 环境发生了改变：比如法律变化和业务变化
    3. 需求基线存在缺陷
    4. 用户变动
    5. 用户对软件的认识变化
    6. 相关产品的出现

4.2. 变更控制过程

1. 以**可控、一致**的方式进行需求基线中需求的变更处理，包括对变化的**评估、协调、批准或拒绝、实现和验证**

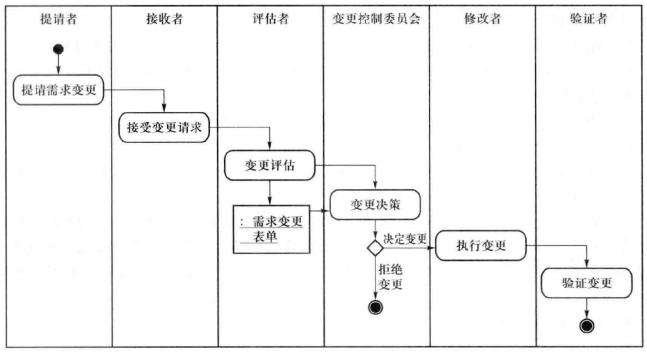


图 17-7　变更控制过程

状态	定义
已验证(verified)	已在集成产品中确认了这一需求的功能实现是正确的。这一需求已经被跟踪到相关的测试用例。这一需求目前可以被认为是已完成了
已删除(deleted)	已批准的需求又从需求基线中取消了。要解释清楚为什么要删除这一需求，以及是谁决定删除的
已否决(rejected)	需求已被提议，但并不在下一版本中实现它。要解释清楚为什么要否决这一需求，以及是谁决定否决的

2. 变更控制过程
    1. 提交需求变化
    2. 评估需求变化可能带来的影响
        1. 利用需求跟踪信息确定变更的影响范围，包括需要修改的系统组件、文档、模型等。
        2. 依据需求依赖信息确定变更将会带来的冲突和连锁反应，确定解决的方法。
        3. 评估变更请求的优先级和潜在风险。
        4. 明确执行变更需要执行的任务，估算变更所需要的工作量和资源。
        5. 评价变更可能给项目计划带来的影响。
    3. 变更评估结果使用正式文档的方式固定，提交给变更控制委员会。
3. 配置管理部门：批准后的变更请求被通知给所有需要修改工作产品的团队成员，由他们完成变更的修改工作。

项目名称：	请求编号：
提请人： 提请理由及优先级： 变更请求描述：	提请日期：
评估人： 评估优先级： 影响范围： 工作量估算： 变更评价：	评估日期： 变更类型：
提交 CCB 日期： CCB 决定：	CCB 决策日期：
修改人： 修改结果：	修改日期：
验证人： 验证结果：	验证日期：
备注：	

图 17-8　变更请求表单

4.3. 变更控制委员会(CCB)

1. 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
2. 变更控制委员会可能由来自下列部门的人员组成
    1. 项目或程序管理部门
    2. 产品管理或者需求分析部门
    3. 开发部门
    4. 测试或者质量保障部门
    5. 市场或客户代表
    6. 编写用户文档的部门

7. 技术支持或帮助部门
8. 配置管理部门

4.4. 注意事项

1. 认识到变更的必要性，并为之制定计划
    1. 定义明确的变更控制过程，建立变更控制的有效渠道
    2. 所有提交的需求变更请求都要进行仔细的评估
    3. 是否进行变更的决定应该由变更控制委员会统一做出
    4. 必须对变更的实现结果进行验证
    5. 需求的变化情况要及时的通知到所有会受到影响的项目涉众
2. 维护需求基线，审计变更记录：保证向项目涉众可以访问到最新的需求和变更情况。
3. 管理范围蔓延
    1. 根据业务目标、产品前景和项目范围，评估每一项提议的新增需求和特性
    2. 对不合理的需求说"不"
4. 灵活应对变更请求
    1. 推迟产品的交付时间
    2. 要求增派人手：在有限的情况下有效
    3. 要求员工加班工作：只能适度的使用
    4. 推迟或者去除尚未实现的优先级较低的需求
    5. 容许产品质量的降低：尽量不使用
5. 使用辅助工具：工具应该具有以下几个特性，以支持需求变更过程：
    1. 可用定义变更请求中的数据项
    2. 可用辅助项目涉众完成变更控制过程中的协作
    3. 可以帮助维护需求基线，审计变更记录
    4. 能够将变更情况及时的通知到相关人员
    5. 可以生成标准的和定制的报告和图表

5. 需求管理的实践调查

5.1. 需求的变更

1. **有效**处理变更非常重要
2. **新增**(Added)需求影响最大
3. **缺陷**修复最为频繁
4. **范围蔓延常**见

5. 需求**可变性**很高
6. 变更控制还需要继续完善

5.2. 需求跟踪

1. 重视和关注了对后向跟踪联系的处理
2. 忽视了对前向跟踪联系的处理
3. 最低层次需求跟踪策略存在广泛
4. 高端用户的需求跟踪实现仍需努力
5. 需求之间的依赖关系困难和复杂
    1. 只有大概20%的需求是完全独立的
    2. 20%左右的需求产生了所有依赖关系的75%。

5.3. 需求管理工具

1. 非常需要需求管理工具
2. 通用的**文本处理器**(Word Processor)和**电子表格**(Spreadsheet)使用最为广泛
3. 部分组织自己开发了专用需求管理工具
4. 很少有组织使用专用的商业需求管理工具
    1. 无法和软件的开发过程以及其他辅助工具进行有效的集成

6. 实例分析

1. 经常出现一个模块的需求刚刚整理完毕或者程序编写了一半，业务已经发生变化的情况。在一年的开发过程中，我们陆续接到的业务变更行政命令多达几十条，这给整个软件开发和推广都带来了很大困惑。为了保证软件正常运行，省局信息中心不得不专门成立了软件推广维护小组，不断就新业务改写程序，各地也不得不在后续的过程中不停的从省局下载新的升级包，好在这件事情已经经历了很多年，大家都已经习惯了。

7. 本章总结

1. 需求管理是发生在需求开发之后的需求工程活动，贯穿于余下的产品生命周期，用于确保需求作业的有效实现
2. 需求管理是一项重要的活动，包括维护需求基线、需求跟踪和需求变更控制
3. 实践调查表明需求管理工作仍然有待人们的努力