book16-需求验证

1. 课程回顾

1. 需求规格说明
    1. 定义用户需求：准确描述需求与其解决方案
    2. 作用：信息传递，拓展记忆，合同协议，指导开发，减少错误，智力资产
    3. 内容：问题-目标-系统特性-用户需求-系统级需求
    4. 优秀的规格说明=模版(剪裁与定制)+写作(引用、强化、术语表、避免干扰文本)

2. 验证(validation)与确认(vertification)

1. 需求验证：以正确的方式建立需求
    1. 需求集是正确的、完备的和一致的
    2. 技术上是可解决的
    3. 它们在现实世界中的满足是可行的和可验证的
2. 需求确认：建立的需求是正确的
    1. 每一条需求都是符合用户原意的

2.1. 需求验证

2.1.1. 概念

1. 验证普遍存在
    1. 获得的用户需求是否正确和充分的支持业务需求?
    2. 建立的分析模型是否正确的反映了问题域特性和需求？细化的系统需求是否充分和正确的支持用户需求?
    3. 需求规格说明文档是否组织良好、书写正确？需求规格说明文档内的需求是否充分和正确的反映了涉众的意图？需求规格说明文档是否可以作为后续开发工作(设计、实现、测试等等)的基础?
2. 需求验证是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

2.1.2. 软件工程中的系统验证

1. **软件测试**：人们最熟知和常用的软件质量保证措施，以考察正在执行的软件的输入/输出或者功能来验证软件的质量。
2. **静态分析**：软件系统的质量保障要求在实际可执行的代码产生之前，要尽可能地依据开发文档、模型或者其他各种可用物件(如原型)进行分析和推理，及早发现错误并进行修正，这些方法统称为静态分析。
3. 验证是贯穿整个软件生命周期的，静态分析和测试是它的最主要手段。
    1. 需求工程中的验证:需求是否正确;是否充分地表达了涉众的需要?
    2. 体系结构设计中的验证:体系结构是否很好地支持了功能需求和非功能需求?
    3. 详细设计中的验证:设计是否遵守体系结构约定?是否为所有的系统功能都设计了方案?
    4. 编码中的验证:代码是否吻合设计?
    5. 测试中的验证:是否所有的需求和系统功能都得到了测试?
    6. 产品/部署中的验证:产品能否在符合要求的环境下正确工作?

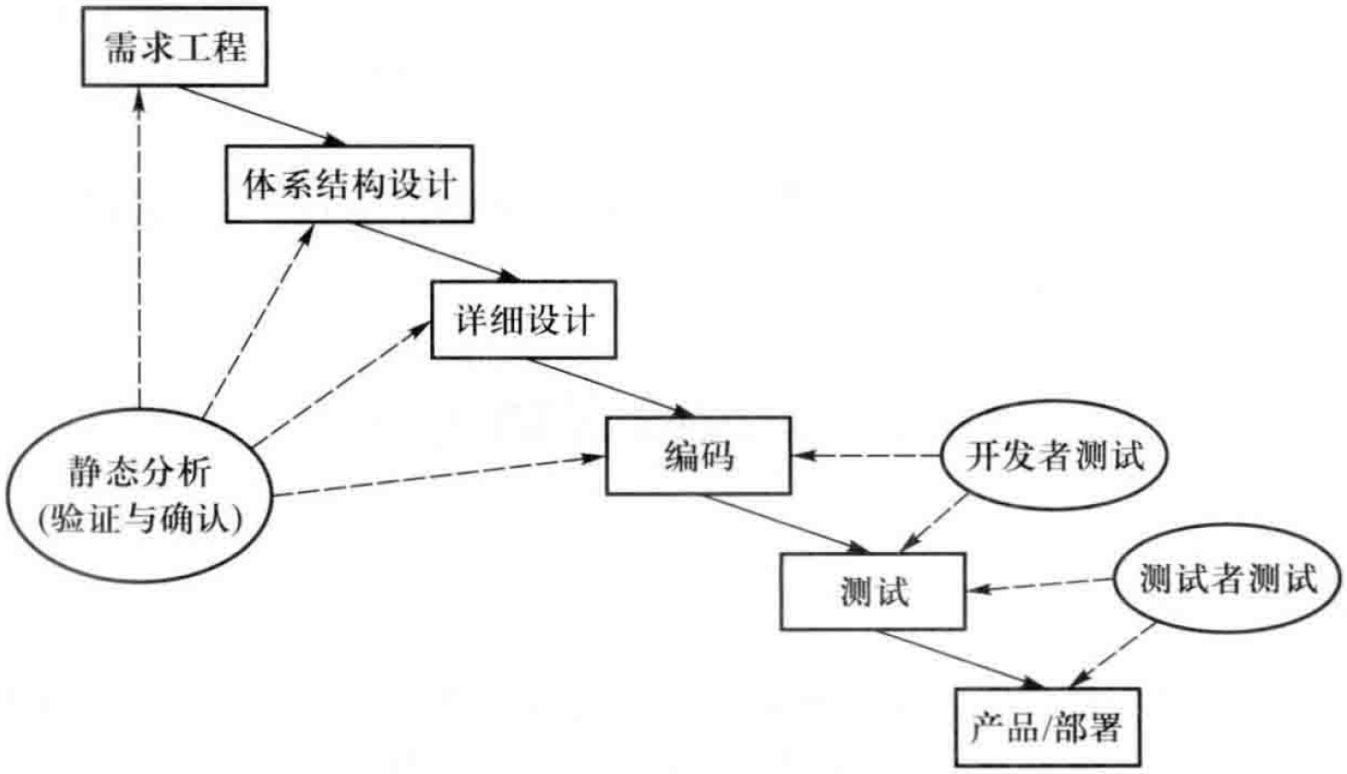

图 16-1　软件开发中的验证与确认活动，源自[Hailpern 2002]

2.1.3. 活动

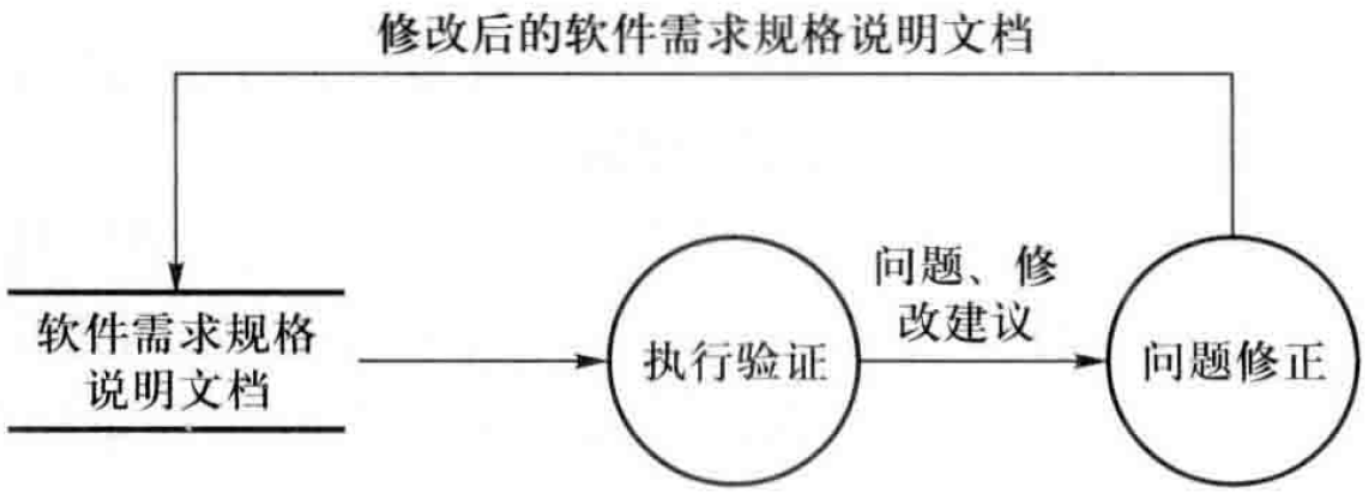

图 16-2　需求验证活动流程图

1. 需求验证并不是一个可以一次结束的活动，它可能需要多次、反复地执行验证。
2. 验证方法：评审、原型与模拟、测试用例开发、用户手册编制、利用跟踪关系和自动化分析。

3. 需求验证方法

3.1. 评审(同级评审)

1. 由作者之外的其他人来检查产品问题的方法
2. 是主要的**静态分析**手段
3. 原则上，每一条需求都应该进行评审

3.1.1. 参与人员

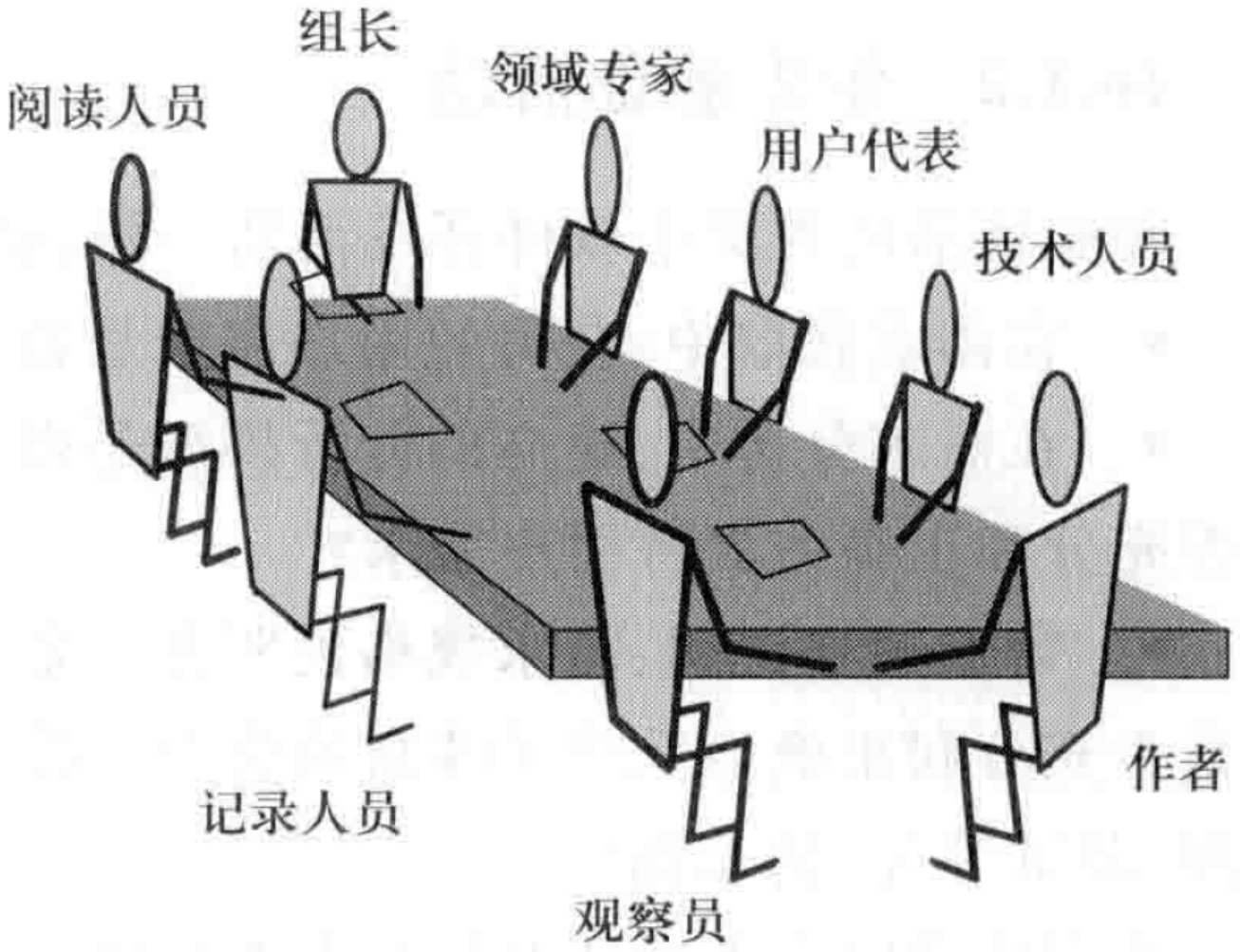

图 16-3　同级评审的参与人员

1. 组织者：负责整个项目活动的组成和规划。
2. 仲裁者：负责确保整个评审过程的正确进行，协调评审活动。主持。
3. 作者：听取评论，在需要时解答评审人员的问题。
4. 阅读人员：负责注意解释软件需求规格说明文档的内容，并且在每次解释后由评审人员指出可能的问题和缺陷。
5. 记录人员：记录评审中发现的问题及修改建议。
6. 收集人员：如果分散评审，则需要收集人员来收集。
7. 评审人员(3-7位)
    1. 领域专家：至少一位
    2. 用户代表：尽可能包含各类型的涉众。
8. 技术人员：需要以被评审文档作为内容开展工作的人。
9. 观察员：对被评审文档内容具有一定经验的人，比如作者同级伙伴等。

3.1.2. 过程

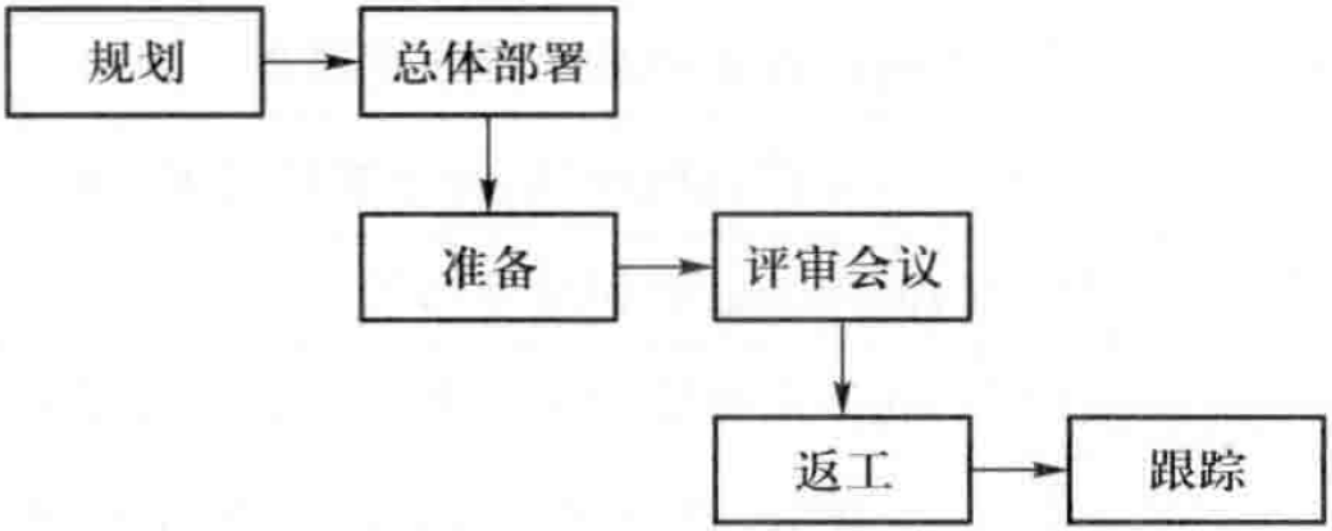

图 16-4　同级评审的过程

1. 规划阶段：作者和仲裁者共同制定评审计划、决定评审会议次数、安排时间、地点、参与人员和评审内容等。
2. 总体部署阶段：作者和仲裁者项所有参加会议的人员描述评审资料的内容、评审的目标以及一些假设，并分发文档。
3. 准备阶段：评审人员各自独立执行评审任务，使用检查清单、场景等方法，将发现的问题记录并交给收集人员或等待开会讨论。
4. 评审会议阶段：通过会议识别、确认、分类发现的错误，会议结束后根据严重程度是否需要修正文档。
5. 返工：作者修改发现的缺陷。
6. 跟踪：仲裁者需要确认所有发现的问题都得到了解决，所有的错误都得到了修正。
7. 评审结束标准：
    1. 评审期间评审人员提出的所有问题都已解决。
    2. 文档中和相关工作产品中的所有更改都已正确完成。
    3. 修订过的文档已经进行了拼写检查。
    4. 所有标识为TBD(待确定)的问题都已经解决,或者已经对每个待确定问题的解决过程、计划解决的目标日期和由谁来解决等编制了文档。
    5. 文档已经在项目的配置管理系统中登记。

3.1.3. 检查方法

表 16-1　常见的检查方法

检查方法	描述
自由方法(ad-hoc)	没有为评审人员提供系统化的引导
检查清单(checklist-based)	以通用的检查清单来引导评审过程
缺陷(defect-based)	用于需求文档,根据缺陷的分类来组织和检查场景
功能点(function point-based)	按照功能点来组织和检查场景
视角(perspective-based)	按照不同涉众类型的视角来组织和检查场景
场景(scenario-based)	对每一个场景,都利用一系列的问题或者细节要求来引导检查过程。缺陷、功能点、视角都是场景方法的一个特例。
逐步提升(stepwise abstraction)	净室软件开发中的一种方法。阅读者描述一些独立代码段的功能,然后将描述的范围逐步扩大,描述的功能抽象逐步提高,直至阅读人员描述了整个评审物件

1. 自由方法和检查清单方法是使用最为广泛的两种方法。
2. 检查清单有一定指导作用。
3. 基于场景方法也是常用的一种方法。
    1. 一些场景提出了评审人员必须回答的关键性问题。
    2. 另一些场景指导评审人员执行工作产品的具体任务。

组织和完整性

- 所有对其他需求的内部交叉引用是否正确?
- 编写的所有需求其详细程度是否一致和合适?
- 需求是否能为设计提供足够的基础?
- 是否确定了每个需求的优先级?
- 是否定义了所有对外的硬件、软件和通信接口?
- 软件需求规格说明中是否包括了所有已知的需求?
- 需求中是否遗漏了必要的信息? 如果有的话,有没有标记为待确定(TBD)?
- 是否对所有预期错误产生的系统行为都编制了文档?

正确性

- 是否有需求与其他需求相冲突或与其他需求重复?
- 是否清晰、简洁、准确地表达了每个需求?
- 是否每个需求都能通过测试、演示、评审或者分析等方法得到验证?
- 是否每个需求都在项目的范围内?
- 是否每个需求都没有内容上和语法上的错误?
- 在现有的资源限制内,是否能实现所有的需求?
- 每一条特定的错误信息是否都是唯一的和具有含义的?

质量属性

- 是否合理地确定了所有的性能目标?
- 是否合理地确定了防护性和安全性方面要考虑的问题?
- 在对质量属性进行了合理的折衷之后,是否对其他相关的质量属性目标已定量的进行了编档?

可跟踪性

- 是否每个需求都具有唯一性并且可以正确地识别?
- 是否每个软件功能需求都可以被跟踪到高层需求?

特殊问题

- 是否所有的需求都是名副其实的需求,而不是设计或实现方案?

图 16-5 需求评审的检查清单,源自[Wiegers 2003]

3.1.4. 类型

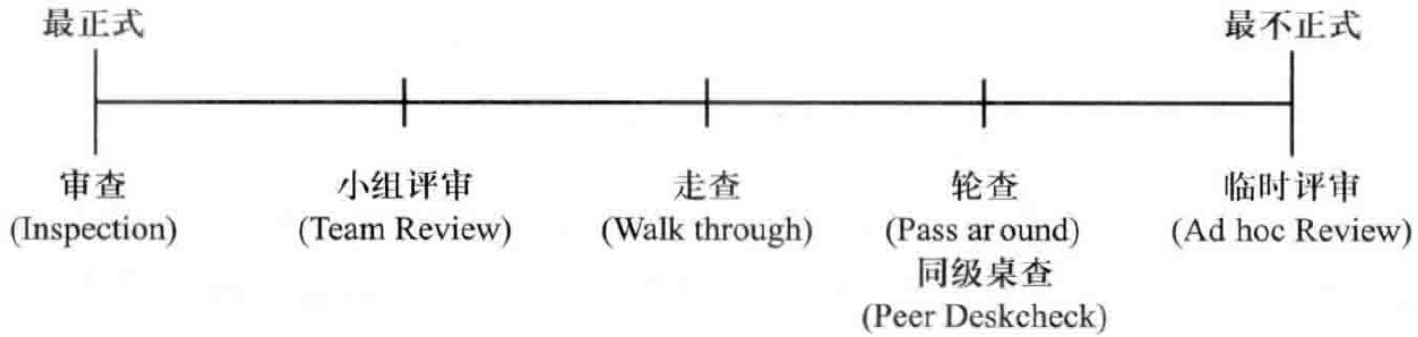

图 16-6 评审的类型,源自[Wiegers 2002]

1. 小组评审是轻型审查

2. 走查：由产品的作者将产品逐一向同事介绍，并希望他们给出意见，只请一个审查人员进行检查。
3. 轮查：同时请作者的多个同事分别进行产品的检查，检查过程中相互讨论，但是最终参与会议讨论的可能只有一部分甚至少数检查人员。
4. 临时评审是最不正式的评审，只是作者临时起意发起的评审活动。

表 16-2　不同评审方法下的活动

评审类型	规划	准备	会议	纠错	跟踪
审查	是	是	是	是	是
小组评审	是	是	是	是	否
走查	是	否	是	是	否
轮查、同级桌查	否	是	可能	是	否
临时评审	否	否	是	是	否

3.2. 原型与模拟

1. 大多数情况下，需求都是在静态的方式下被加以验证，如评审方法。
2. 涉及到复杂的动态行为时，可能需要使用原型或模拟方法来加以验证。
3. 成本较高，只适用于验证一些静态方法力所不及的复杂需求。

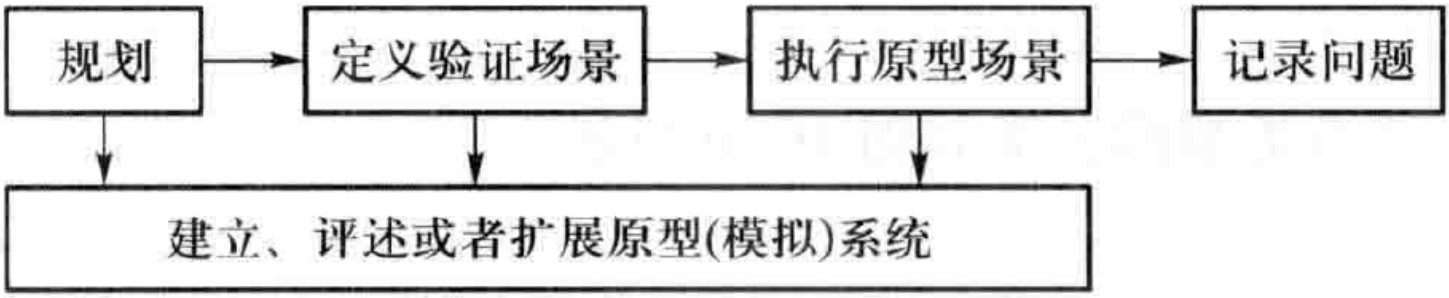

图 16-7　需求验证的原型方法

3.3. 开发测试用例

1. 如果无法为某条需求定义完备的测试用例，那么它可能就存在着**模糊、信息遗漏、不正确**等缺陷
2. 例外(没有问题的需求)
    1. 排斥性需求(Exclusive Requirements)：这种需求要求特定的行为绝对不会发生，例如需求可能会要求系统故障不能导致数据库的崩溃
    2. 全局性非功能性需求(Global Non-Functional Requirements)：例如可靠性、可用性等，对这些需求的测试往往都是大数据集的处理

3.3.1. 开发系统测试用例

1. 以需求为线索，开发测试用例套件
2. 使用测试技术确定输入/输出数据，开发测试用例

3.3.2. 测试用例套件

1. 基于用例描述，可以为销售处理确定测试用例套件

测试用例套件	覆盖流程							
TUS1	正常流程		3a	3b	5-8a			
TUS2	正常流程	1a					9a	11a
TUS3	正常流程					5-8b		

3.3.3. 建立测试用例

1. 主要是基于规格的技术，设计测试场景的输入与输出数据

ID	输入				预期输出
	商品信息	特价	赠品	支付	
TUS1-1	无商品	无	无	无	不做任何处理，关闭销售任务
TUS1-2	商品1（1、1(双)、35） 商品2（2、1(双)、50）	无	无		
TUS1-3	商品1（1、1(双)、35） 商品2（2、1(双)、50）	无	无	85	无找零，系统行为满足后置条件
TUS1-4	商品1（1、1(双)、35） 商品2（2、1(双)、50）	商品1特价20		100	找零30，系统行为满足后置条件
TUS1-5	商品1（1、1(双)、35） 商品2（2、1(双)、50）	总额特价50以上0.8折		100	找零31，系统行为满足后置条件
TUS1-6	商品1（1、1(双)、35） 商品2（2、1(双)、60）	商品1特价20 总额特价50以上0.8折		100	找零32，系统行为满足后置条件

3.4. 用户手册编制

1. 验证功能需求：对软件系统**功能和实现**的描述
2. 验证项目范围：对系统**没有实现**的功能的描述
3. 验证异常流程需求：**问题和故障**的解决
4. 验证环境与约束需求：系统的**安装和启动**

3.5. 利用跟踪关系

1. 业务需求->用户需求->系统需求：如果业务需求和用户需求没有得到后项需求(用户需求和系统需求)的充分支持，那么软件需求规格说明文档就存在不完备的缺陷。
2. 系统需求->用户需求->业务需求：如果不能依据跟踪关系找到一条系统需求的前项用户需求和前项业务需求，那么该需求就属于非必要的需求。

3.6. 自动化分析

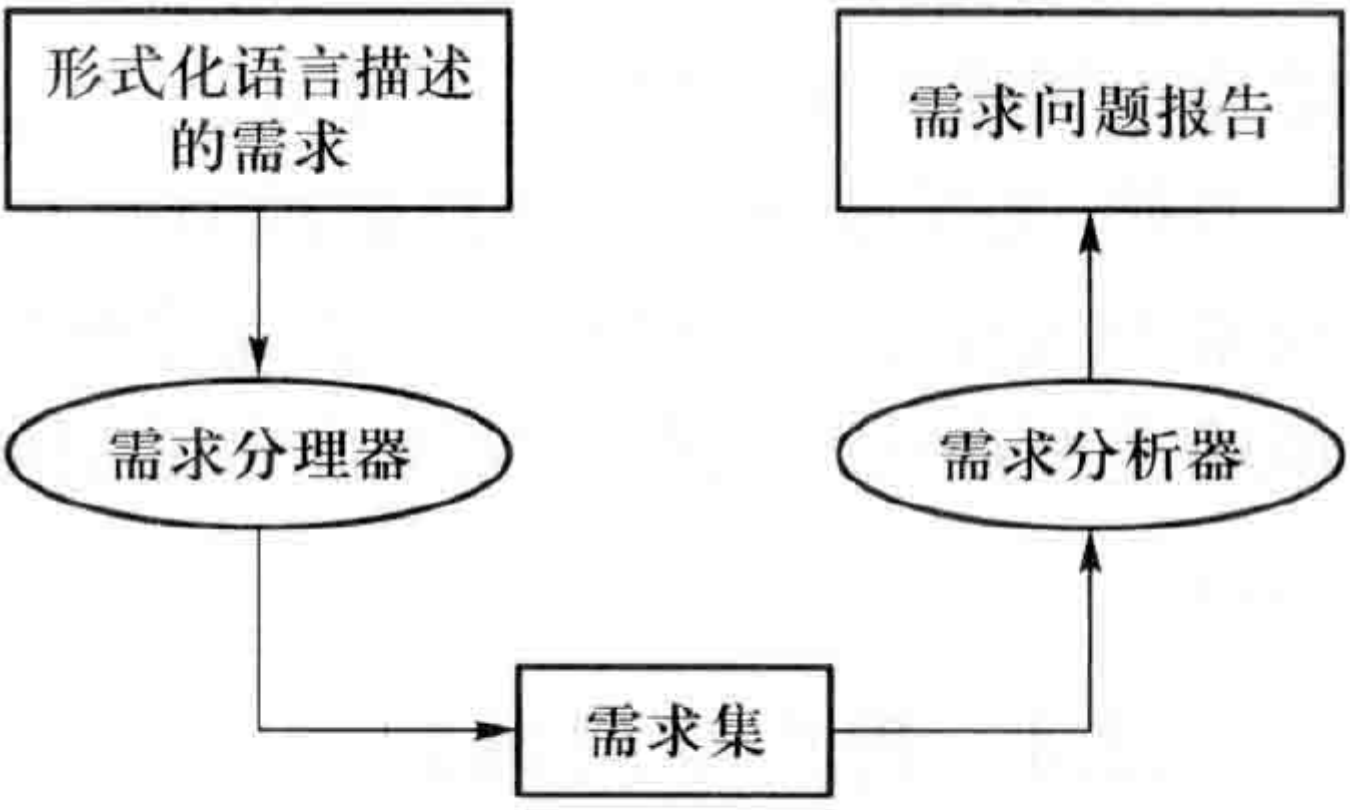

图 16-8　需求文档的自动化分析

4. 问题修正

1. 需求澄清(Requirements Clarification)
    1. **理解偏差**：重新进行分析工作
    2. **分析遗漏**：重新分析和文档化这部分信息
    3. **表达不当**：重新以合适的方式表达
2. 缺失需求：重新执行需求获取等一系列工作
3. 需求冲突：协商解决
4. 不切实际的期望：项目调整与需求协商

5. 需求验证的实践调查

1. 需求验证是重要的
2. 需求验证是容易被忽视的
3. 需求验证的方法是多样的
    1. 评审和原型最为广泛
    2. 客户对线索(Threads)和场景(Scenarios)表现出了最大的兴趣
    3. 技术人员、领域专家、客户以及用户是最合适的评审者

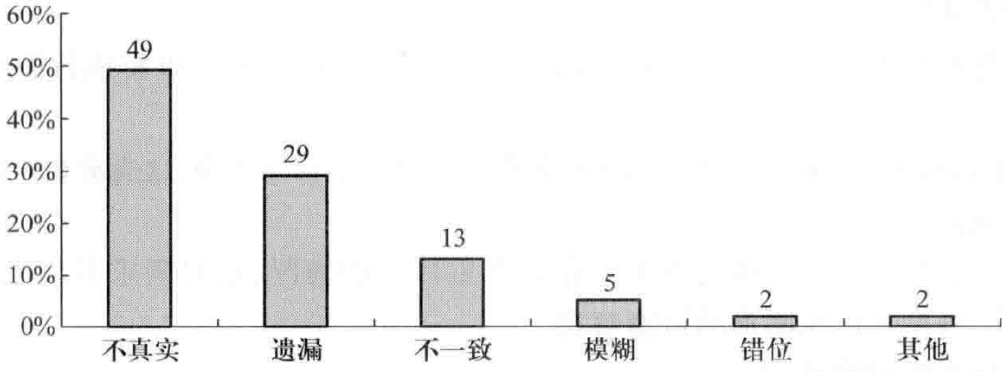

图 16-9　需求定义的常见错误，数据源自于[Young 2002]

表 16-3　NASA 项目中发现的需求缺陷，数据源自[Hayes 2003]

主要缺陷	百分比	主要缺陷	百分比
不完整	20.9	不可跟踪	1.4
遗漏	32.9	不可验证	0.5
不正确(不真实)	23.9	错位	0.7
模糊	6.1	有意偏离(Intentional Deviation)	0.7
不可行	1.4	冗余或重复	0.5
不必要(Over Specification)	6.3		

6. 示例分析

1. 需求虽然写好了也定稿了，但是并没有得到最终确认就开始了软件开发工作。这种现象主要是由于业务小组和技术小组沟通不全面造成的，在双方就某一问题产生分歧的情况下，没有一个能出来拍板的人决定(有权利决定的领导不参与开发和需求编写)。所以整个项目的开发是在业务小组和技术小组的争论中走过的。经常出现业务小组提出的方案技术小组难以落实，等到后期变通修改造成功能损失的情况。因为需求得不到最终确认，一直在修改中，造成技术小组不停的修改已经编写完毕的模块，有些改动甚至涉及到公共基类的修改和各模块之间的关联，造成很大的浪费。
2. 系统开发过程中，没有好的办法检测需求落实的情况。税务系统中专业性很强，经常出现业务人员不懂计算机的情况，有些业务人员甚至不会上网。技术小组编写的代码是否已经实现了全部功能，很多业务人员在测试过程中发现不了问题，造成最后验收的时候功能是否实现由技术小组说了算。

7. 本章小结

1. 验证与确认是软件工程当中一项重要的活动。需求验证是需求工程中发生的对需求规格说明文档进行的验证与确认活动
2. 需求验证有多种有效的方法，实践中最为重要和广泛应用是的评审方法和原型方法
3. 需求验证不仅要发现问题，而且要监督问题的解决