第三部分　软件需求管理

第16章　需求管理的原则与实现

在第一章中，我们将需求工程分为需求开发和需求管理。需求开发包括对一个软件项目需求的获取、分析、 规格说明及验证。典型需求开发的结果应该有项目视图和范围文档、使用实例文档、软件需求规格说明及相关分析模型。经评审批准，这些文档就定义了开发工作的需求基线（baseline）。这个基线在客户和开发人员之间就构筑了计划产品功能需求和非功能需求的一个约定（agreement）。工程项目可能会有其它的约定，例如可交付性、约束条件、进度安排、预算及合同约定等。但这些均超出了本书范围。

需求约定是需求开发和需求管理之间的桥梁，需求管理包括在工程进展过程中维持需求约定集成性和精确性的所有活动，如图 16-1所示。需求管理强调：

- 控制对需求基线的变动。
- 保持项目计划与需求一致。
- 控制单个需求和需求文档的版本情况。
- 管理需求和联系链之间的联系或管理单个需求和其它项目可交付品之间的依赖关系。
- 跟踪基线中需求的状态。

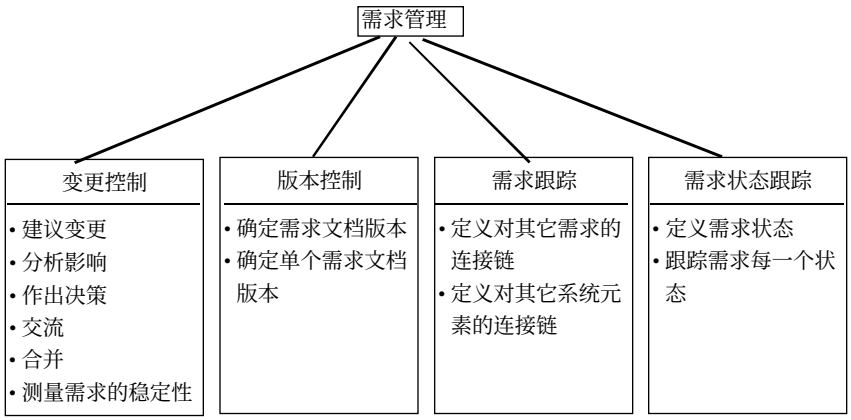

图16-1　需求管理的主要活动

本章给出了需求管理的基本原则。第三部分的其它内容更详细地评述了专门的需求管理的策略：包括变更控制（第 17章），需求跟踪（第 18章）和变更的影响分析（第 18章）。第三部分的最后是关于帮助管理项目需求的商业工具的讨论（第 19章）。

16.1　需求管理和过程能力成熟度模型

过程能力成熟度模型（Capability Maturity Model，CMM）对需求管理是一个有用的指导

（CMU/SEI 1995）。位于宾夕法尼亚洲匹兹堡市的卡内基梅隆大学所属的软件工程研究所提出了软件过程能力成熟度模型的概念。 CMM是在软件开发机构中被广泛地用来指导过程改进工作的模型。该方法描述了软件处理能力的五个成熟级别。处于一级的组织典型地以非正式的方式管理项目进度，要获得成功，主要依靠天才从业者和管理者的英雄史诗般的奋斗。处于更高成熟度级别的组织把具有创造性、训练有素的员工同软件工程和项目管理过程结合起来，将持续不断地获得成功。

为达到软件过程能力成熟度模型的第二级，组织必须具有在软件开发与管理的六个关键过程域（key process areas，KPA）以展示达到目标的能力。需求管理是其中之一，它的目标如下：

1) 把软件需求建立一个基线供软件工程和管理使用。

2) 软件计划，产品和活动同软件需求保持一致。

无论是否知道或关心过程成熟度模型，大多数软件开发组织将会从达成这两个目标中获益。过程成熟度模型确定若干先决条件和技术策略，使组织能持续地达到这两个目标，但并不指定组织必须遵循的需求管理过程。

需求管理的关键过程领域不涉及收集和分析项目需求。而是假定已收集了软件需求或已由更高一级的系统给定了需求。一旦需求到手且文档化了，软件开发团队和有关的团队（例如质量保证和测试）需要评审文档。发现问题应与客户或其它需求源协商解决，软件开发计划是基于已确认的需求。

开发团队在向客户、市场部或经理们作出承诺（ commitment）之前，应该确认需求和确认约束条件、风险、偶然因素、假定条件。也许不得不面对由于技术因素或进度原因而不现实的需求作出承诺。但是，决不要承诺任何无法实现的事。

关键处理领域同样建议通过版本控制和变更控制来管理需求文档。版本控制确保随时能知道在开发和计划中正在使用的需求的版本情况。变更控制提供了支配下的规范的方式来统一需求变更，并且基于业务和技术的因素来同意或反对建议的变更。当在开发中修改、增加、减少需求时，软件开发计划应该随时更新以与新的需求保持一致。不反映现实的计划于事无补。

当接受了所建议的变更时，你可能在进程调度或质量上不能满足这项变更。在这种情况下，必须就约定的变更与所涉及的经理、开发者以及其它相关组织进行协商。通过如下方法能使项目反映最新的或变更过的需求。

- 暂时搁置次要需求。
- 得到一定数量的后备人员。
- 短期内带薪加班处理。
- 将新的功能排入进度安排。
- 为了保证按时交工使质量受些必要的影响（通常，这是缺省反应）。

由于项目在特性、进度、人员、预算、质量各个方面的要求不同，所以不存在一个放之四海皆准的模式（Wiegers 1996a）。根据早期计划阶段中项目风险承担者确定的优先级顺序挑选各项选择。不管你对变更需求或项目情况（如全体职工工作完成量）采取何种措施，必要时调整一些约定仍是需要养成的一个好习惯。这总比不现实地期待所有新要求在原定交付日期前魔术般地实现且其他方面（例如预算或员工工作强度）不受什么影响要好。

即使你现在没用CMM来指导你的软件过程改进，以上所述关于需求管理关键过程领域内的原则和策略（practice）总是有用的。每个开发组织都会从这些方法应用中获益。

16.2 需求管理步骤

开发组织应该定义项目组执行管理他们需求的步骤。文档化编写这些步骤能使组织成员持续有效地进行必要的项目活动。请考虑选择以下主题：

- 用于控制各种需求文档和单个需求版本的工具、技术和习惯做法。
- 建议、处理、协商、通告新的需求和变更给有关的功能域的方法。
- 如何制定需求基线。
- 将使用的需求状态，并且是谁允许作出的变更。
- 需求状态跟踪和报告过程。
- 分析已建议变动的影响应遵循的步骤。
- 在何种情况下需求变更将会怎样影响项目计划和约定。

你可以在一个文档中包含上面所有的信息。或者，你可能喜欢专题分述，例如分成变更控制过程，影响分析过程，状态跟踪过程。这些过程可能在多个项目中都有用，因为他们反映每个项目所应遵循的公共功能。

16.3 需求规格说明的版本控制

“我终于实现了库存报告中重排序的功能。”Shari在项目的每周例会上说。

“噢，用户在两周前就取消这个功能了。”项目管理者说，“你没看改过的软件需求规格说明吗？”

如果以前曾听过这样的谈话，你一定知道浪费时间为已废弃的需求工作会使员工多么地沮丧。我了解到曾有一个开发组在把改进的软件版本交去测试后，收到一大堆错误发现报告。其实是测试者使用了一个已过时软件需求规格说明，结果导致了一大堆错误。开发组花了大量的时间试着确定问题，甚至花了相当多时间对照正确版本的软件需求规格说明进行重复测试。

版本控制是管理需求的一个必要方面。需求文档的每一个版本必须被统一确定。组内每个成员必须能够得到需求的当前版本，必须清楚地将变更写成文档，并及时通知到项目开发所涉及的人员。为了尽量减少困惑、冲突、误传，应仅允许指定的人来更新需求。这些策略适用于所有关键项目文档。

每一个公布的需求文档的版本应该包括一个修正版本的历史情况，即已做变更的内容、变更日期、变更人的姓名以及变更的原因。你应该使用标准修改符，例如，中划线代表取消，下化线代表添加，在页边空白的竖划线指示每个变动的位置。因为这些符号会弄乱文档，支持修改符的字处理软件使你能够预览和打印编辑后的文档或最终的结果。可以考虑给每个需求标记上版本号，当修改需求后就增加版本号。

版本控制的最简单方法是根据标准约定手工标记软件需求规格说明的每一次修改。根据修改日期或印刷日期区别文档的不同版本容易产生错误，所以不被推荐。使用一种手工方法，任何新的文档的第一版当标记为“1.0版（草案1）”，下一稿标记为“1.0版（草案2）”，在文档被采纳为基线前，草案数可以随着改进逐次增加。而当文档被采纳后被标记为“1.0正式版”。

若只有较小的修改，可认为是“ 1.1版（草案1）”。若有较大的修改时，可认为是“ 2.0版（草案1）”（“ 较大 ”和“ 较小 ”只是一个主观的判断）。这个方式清楚地区分草稿和定稿的文档版本，但要求用手工来操作。

一个具有更高级别的版本控制包括用版本控制工具来存储需求文档，例如用登录（ check-in）和检出（check-out）程序来管理源代码。这方面有很多商业配置管理工具。我知道一个项目用诸如Microsoft Word这样的工具管理了几百个使用实例文档。这个工具能让每个组员得到每个使用实例文档的先前版本，同时提供一个记录每一个文档变更的日志。对存储在工具中的文档，项目需求分析人员具有对它可以读和写的权利，而其他成员只能读。这个工具在该项目的版本控制计划中工作得很好。

版本控制的最有力方法是用一个商业需求管理工具的数据库存储需求（详情见 19章）。这些工具能跟踪和报告每个需求的变动历史，当你需要恢复早期的需求时这很有价值。在添加、变动、删除、拒绝一个需求后，附加一些评语描述变更的原因在将来需要讨论时将会很有用。

16.4 需求属性

除了文本，每个功能需求应该有一些相关的信息或称之为属性与之相联系。这些属性在它的预期功能性之外为每个需求建立了一个上下文和背景资料。属性值可以写在一张纸上，存储在一个数据库或需求管理工具中。商业工具除由系统产生一些属性外，还可以由你自己定义各种数据类型的其它属性。这些工具允许过滤、排序、查询数据库来察看按选择的需求属性的需求集。

对大型的复杂项目来说，丰富的属性类别显得尤为重要。在你的每个需求文档中考虑明确如下的属性：

- 创建需求的时间
- 需求的版本号
- 创建需求的作者
- 负责认可该需求的人员
- 需求状态
- 需求的原因或根据（或信息的出处）
- 需求涉及的子系统
- 需求涉及的产品版本号
- 使用的验证方法或接受的测试标准
- 产品的优先级或重要程度（例如高、中、低或把你能定义的属性来表示成本书第 13章所描述的优先级的四个方面：收益、损失、成本、风险）
- 需求的稳定性（在将来需求可能变更的指示器，不稳定的需求意味你应给予较多的关注，因为你将面临不定的、混沌的、或不能重复的业务过程。）

定义和更新这些属性值是需求管理成本的一部分。精心挑选属性最小子集能帮助你有效管理项目。例如，你不必把负责认可需求的人员和需求涉及的子系统都记录下来。如果这样的属性在别的地方已经设置了，在总的开发跟踪系统中就不必在需求数据库中重复设置。

在整个开发过程中，跟踪每个需求的状态是需求管理的一个重要方面（ Caputo 1998）。在每一可能的状态类别中，如果你周期性地报告各状态类别在整个需求中所占的百分比将会改

进项目的监控工作。假如你有清晰的要求，指定了允许修改状态信息的人员和每个状态变更应满足的条件，跟踪需求状态才能正常工作。工具能帮你跟踪每次状态改变的日期。

表16-1建议了几种需求状态。一些专业人员添了一些状态如“已设计”（表明功能需求部分已设计和评审出来了）或“已交付”（意味着修改后的软件已交付用户进行 β版测试），保存一份已提出但没有批准的（即被拒绝状态）需求的记录是非常有用的，因为在开发中这些需求随时会被重新提出。

表16-1 建议的需求状态表

状 态 值	定 义
已建议	该需求已被有权提出需求的人建议
已批准	该需求已被分析，估计了其对项目余下部分的影响（包括成本和对项目其余部分的干扰），已用一个确定的产品版本号或创建编号分配到相关的基线中，软件开发团队已同意实现该项需求
已实现	已实现需求代码的设计、编写和单元测试
已验证	使用所选择的方法已验证了实现的需求，例如测试和检测，审查该需求跟踪与测试用例相符。该需求现在被认为完成
已删除	计划的需求已从基线中删除，但包括一个原因说明和做出删除决定的人员

图16-2以图示的方法跟踪了一个假想为期 10个月的项目持续过程中的需求状态，展示了在每个月底各个状态的系统需求的百分比。这个图并不能表示基线中的需求数量是否正在随时间而改变。但它说明了你是如何达到完全验证所有已获批准需求这个目标的。

对这些需求进行分门别类检测要比对每个需求的完成情况都检测要现实一些。软件开发者报告完成任务的百分比时，往往会过分乐观，常常对未完工的需求拥有较大的信心。趋向于“集中”（round up），他们的进展将导致软件项目或主要任务在很长的时间内处在百分之九十完成的状态这种情况。只有当指定的转换条件满足时才能改变需求的状态。某个状态的改变会导致更新需求跟踪能力矩阵，该矩阵指示与该需求相关的设计、代码、测试元素。这些将会在18章论述。

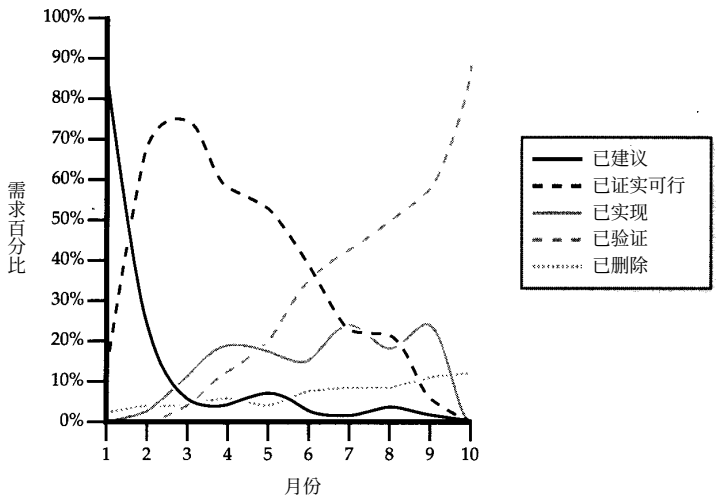

图16-2 在整个项目开发周期中跟踪需求状态的分布

16.5 度量需求管理的效果

在每个项目的工作分类细目结构中需求管理活动应该表现为分配有资源的任务。测算当前项目中的需求管理成本，是计划未来需求管理工作或经费的最佳途径。

一个从未度量过工程任何一个方面的组织通常发现很难开始保持一个耗时记录。测算实际开发和项目管理的工作量要求一个文化上的改变和养成记录日常工作的习惯。然而，测算并不像人们所担心的那样花费时间，了解成员花费在各个项目任务上的确切工作量会使你获得有价值的资料（Wiegers 1996a）。应该注意到工作量计算与翻过的日历时间不成正比，任务进度可能被打断或因同客户协商造成拖延。每个单元的工作时数总和表明一个任务的工作量，这个数据没必要随外界因素变化，但总体上却要比原计划长一些。

跟踪实际的需求管理效果能使你了解组员是否采取了措施进行需求管理。执行需求管理措施不得力，会由于不受约束的变更、范围延伸和遗漏需求等原因而增加项目的风险。考虑一下需求管理的下列活动的效果：

- 提出需求变更和已建议的新需求。
- 评估已建议的变更，包括影响分析。
- 变更控制委员会活动。
- 更新需求文档或数据库。
- 在涉及人员或团队中交流需求的变更。
- 跟踪和报告需求状态。
- 定义和更新需求跟踪能力信息。

尽管忽视和效率不高会随时发生，管理项目需求能确保你投资到需求的收集、归档和分析的努力没有白费。有效的需求管理策略能在整个开发过程中使项目参与者获悉需求的当前状态信息，从而减少大家在需求认识上的差距。

下一步：

- 恰当选择用来描述功能需求生存期的状态。在软件需求规格说明中定义每个需求的当前状态并且在开发过程中不断更新它。
- 定义一个确认需求文档的版本控制计划， 把这个计划编写成文档并作为需求管理过程的一部分。
- 把组织用来管理每个项目需求的步骤描述下来。让几个分析员来起草、评审、实验并最终确定过程活动以保证可交付性。确保选择的过程步骤是实际可行和现实的，并且对这个项目有所帮助。