第3章　需求工程的推荐方法

10年前，我曾热衷于追求软件的开发方法论的研究，将整套整套的模型、技术等用于解决项目难题。但现在我更注重应用“最佳方法”。最佳方法强调将软件工具包拆分成多个子包以分别应用于不同的问题，而并不是去设计或购买一整套的解决方案。即使你采用了一套商业上的方法，你也应当在其中增加那些在业界被认为行之有效的推荐技术。

“最佳方法”这个词值得讨论一下：谁能确定什么是“最佳”的呢？而且，得到这个结论的依据何在？一种方案是把这方面的专家召集起来分析众多不同组织中成功和失败的项目（Browm 1996）。专家们将那些成功项目中提供高效的方法和失败项目中导致低效甚至无效的方法都归纳出来。这样，专家们就能找到公认的能收到实效的关键方法。这些方法即是“最佳方法”，其本质就是有助于项目成功的有效方法。

本章的标题是“需求工程的推荐方法”而非“最佳方法”。下面分七类介绍了四十余种方法，能有助于开发小组做好需求工作，推荐方法如表 3-1所列。

表3-1　需求工程推荐方法

知识技能	需求管理	项目管理
• 培训需求分析人员 • 培训用户代表和管理人员 • 培训应用领域的开发人员 • 汇编术语	• 确定变更控制过程 • 建立变更控制委员会 • 进行变更影响分析 • 跟踪影响工作产品的每项变更 • 编写需求文档的基准版本和控制版本 • 维护变更历史记录 • 跟踪需求状态 • 衡量需求稳定性 • 使用需求管理工具	• 选择合适的生存周期 • 确定需求的基本计划 • 协商约定 • 管理需求风险 • 跟踪需求工作

需求开发

获　取	分　析	编写规格说明书	验　证
• 编写项目视图与范围 • 确定需求开发过程 • 用户群分类 • 选择产品代表 • 建立核心队伍 • 确定使用实例 • 召开应用程序开发联系（JAD）会议 • 分析用户工作流程 • 确定质量属性 • 检查问题报告 • 需求重用	• 绘制关联图 • 创建开发原型 • 分析可行性 • 确定需求优先级 • 为需求建立模型 • 编写数据字典 • 应用质量功能调配（QFD）	• 采用软件需求规格说明模版 • 指明需求来源 • 为每项需求注上标号 • 记录业务规范 • 创建需求跟踪能力矩阵	• 审查需求文档 • 依据需求编写测试用例 • 编写用户手册 • 确定合格的标准

并非上面所有的条目都是最佳方法，或许也并非全部经过了系统地评估。但无论如何，我和许多实践者都觉得这些技术是很有效的 (Sommerville and Sawyer 1997)。其中每一条都在本章给予了简要介绍，并在其他章节或其他更详细地讨论技术来源的地方给出了参考文献。

表3-2把表3-1中的方法按实施的优先顺序和实施难度进行了分组。由于所列的方法都是有所裨益的，故最好是循序渐进，先从那些相对容易实施而对项目有很大影响的方法开始。

表3-2 实施需求工程的推荐方法

优先级别	难度		
	高	中	低
高	• 确定需求开发过程 • 确定需求的基本计划 • 协商约定	• 确定使用实例 • 确定质量属性 • 确定需求优先级 • 采用软件规格说明模板 • 确定变更控制过程 • 建立变更控制委员会 • 审查需求文档	• 培训应用领域的开发人员 • 编写项目视图与范围 • 用户群分类 • 绘制关联图 • 指明需求来源 • 为每项需求注上标号 • 编写需求文档的基准版本和控制版本
中	• 培训用户代表和管理人员 • 为需求建立模型 • 管理需求风险 • 使用需求管理工具 • 创建需求跟踪能力矩阵	• 培训需求分析人员 • 建立核心队伍 • 创建开发原型 • 分析可行性 • 确定合格的标准 • 进行变更影响分析 • 跟踪影响工作产品的每项变更 • 选择合适的生存周期	• 汇编术语 • 选择产品代表 • 编写数据字典 • 记录业务规范 • 依据需求编写测试用例 • 跟踪需求状态
低	• 召开应用程序开发联系（JAD）会议 • 需求重用 • 应用质量功能调配（QFD） • 衡量需求稳定性	• 分析用户工作流程 • 检查问题报告 • 编写用户手册 • 维护变更历史记录 • 跟踪需求工作	

不要想着把所有这些方法都用于你的下一个项目。而应该考虑将其中的一些方法推荐到你的需求工具箱中。不管你的项目处在开发的哪个阶段，你都可以马上开始应用某些方法，譬如变更管理的处理。其它如需求获取等可以在你的下一个项目开始时付诸应用。当然其它一些方法也可能并不适合你目前的项目。

第4章介绍一些用于评价需求工程的方法，并可设计一张实施需求方法改进的步骤图。具体的改进方法在此处和第4章中都给予介绍。

3.1 知识技能

绝大部分的软件开发人员都没有接受过高效需求工程所需技能的正规培训。但许多开发人员在职业生活中的某个阶段总会扮演一个需求分析员的角色，与客户一起工作：收集，分析，编写需求文档。不能过高期望开发人员在需求工程的信息沟通中的“天份”。一定的培训将有助于提高需求分析员的能力和水平。

因为需求对项目成功极为重要，所有的项目风险承担者都应该对需求工程的重要性、合理性及其方法有一个基本的了解。把项目风险承担者（例如开发人员，市场人员，客户，测试人员和管理人员）召集起来进行为期一天的需求过程概要学习，这对建立一个合作团队是很有效的。所有参与者都会更好地明白各自所面临的挑战是什么，以及为了整个团队获得成功大家都需要作些什么。同样，开发人员也能对应用领域的术语和一些基本概念有大致的了解。

1) 培训需求分析人员　所有的开发人员都应接受一个基本的需求工程培训。但那些负责收集(capturing)、编写文档和分析用户需求的人员应当进行为期一周或更长时间的培训。把高水平的需求人员组织起来，通过良好的信息交流，了解应用领域并有效地应用需求工程中的成熟技术。

2) 培训软件需求的用户代表和管理人员　参与软件开发的用户代表应接受为期一天左右关于需求工程的培训，开发管理者和客户管理者也应参加。这样的培训将使他们明白强调需求的重要性，以及忽略需求带来的风险。参加过我组织的需求讨论会的一些用户表示，他们在此之后更能理解软件开发人员了。

3) 让开发人员了解应用领域的基本概念　组织一些简短的关于客户业务活动、术语、目标等方面的讨论会以帮助开发人员对应用领域有个基本了解。这能减少误解及工程中的返工。你可能要为每位开发人员安排一个用户伙伴以便在项目过程中解释业务术语和概念。产品代表就应该扮演这样的角色。

4) 编写项目术语汇编　为减少沟通方面的问题，编一部术语汇编将项目应用领域的专用词汇给予定义说明，既要包括那些有多种含义与用法的术语，也要包括那些在专用领域和一般使用中有不同含义的词。

3.2 需求获取

第1章讨论了需求的三个层次：业务，用户和功能。在项目中它们在不同的时间来自不同的来源，也有着不同的目标和对象，并需以不同的方式编写成文档。业务需求（或产品视图和范围）不应包括用户需求（或使用实例），而所有的功能需求都应该源于用户需求。同时你也需要获取非功能需求，如质量属性。你可以在下列章节中找到相关主题的详细内容：

- 第4章 —— 确定需求开发过程。
- 第6章 —— 编写项目视图和范围文档。
- 第7章 —— 将用户群分类并归纳其特点，为每个用户类选择产品代表（product champion）。
- 第8章 —— 让用户代表确定使用实例。
- 第11章 —— 确定质量属性和其它非功能需求。

1) 确定需求开发过程　确定如何组织需求的收集、分析、细化并核实的步骤，并将它编写成文档。对重要的步骤要给予一定指导，这将有助于分析人员的工作，而且也使收集需求活动的安排和进度计划更容易进行。

2) 编写项目视图和范围文档　项目视图和范围文档应该包括高层的产品业务目标，所有的使用实例和功能需求都必须遵从能达到的业务需求。项目视图说明使所有项目参与者对项目的目标能达成共识。而范围则是作为评估需求或潜在特性的参考。

3) 将用户群分类并归纳各自特点　为避免出现疏忽某一用户群需求的情况，要将可能使

用产品的客户分成不同组别。他们可能在使用频率、使用特性、优先等级或熟练程度等方面都有所差异。详细描述出它们的个性特点及任务状况，将有助于产品设计。

4) 选择每类用户的产品代表　为每类用户至少选择一位能真正代表他们需求的人作为那一类用户的代表并能作出决策。这对于内部信息系统的开发是最易实现的，因为此时，用户就是身边的职员。而对于商业开发，就得在主要的客户或测试者中建立起良好的合作关系，并确定合适的产品代表。他们必须一直参与项目的开发而且有权作出决策。

5) 建立起典型用户的核心队伍　把同类产品或你的产品的先前版本用户代表召集起来，从他们那里收集目前产品的功能需求和非功能需求。这样的核心队伍对于商业开发尤为有用，因为你拥有一个庞大且多样的客户基础。与产品代表的区别在于，核心队伍成员通常没有决定权。

6) 让用户代表确定使用实例　从用户代表处收集他们使用软件完成所需任务的描述 —— 使用实例，讨论用户与系统间的交互方式和对话要求。在编写使用实例的文档时可采用标准模版，在使用实例基础上可得到功能需求。

7) 召开应用程序开发联系会议　应用程序开发联系（JAD）会议是范围广的、简便的专题讨论会（workshop），也是分析人员与客户代表之间一种很好的合作办法，并能由此拟出需求文档的底稿。该会议通过紧密而集中的讨论得以将客户与开发人员间的合作伙伴关系付诸于实践（ Wood and Silver 1995）。

8) 分析用户工作流程　观察用户执行业务任务的过程。画一张简单的示意图（最好用数据流图）来描绘出用户什么时候获得什么数据，并怎样使用这些数据。编制业务过程流程文档将有助于明确产品的使用实例和功能需求。你甚至可能发现客户并不真地需要一个全新的软件系统就能达到他们的业务目标（ McGraw and Harbison 1997）。

9) 确定质量属性和其它非功能需求　在功能需求之外再考虑一下非功能的质量特点，这会使你的产品达到并超过客户的期望。这些特点包括性能、有效性、可靠性、可用性等，而在这些质量属性上客户提供的信息相对来说就非常重要了。

10) 通过检查当前系统的问题报告来进一步完善需求　客户的问题报告及补充需求为新产品或新版本提供了大量丰富的改进及增加特性的想法，负责提供用户支持及帮助的人能为收集需求过程提供极有价值的信息。

11) 跨项目重用需求　如果客户要求的功能与已有的产品很相似，则可查看需求是否有足够的灵活性以允许重用一些已有的软件组件。

3.3 需求分析

需求分析（requirement analysis）包括提炼、分析和仔细审查已收集到的需求，以确保所有的风险承担者都明白其含义并找出其中的错误、遗漏或其它不足的地方。分析员通过评价来确定是否所有的需求和软件需求规格说明都达到了第 1章中优秀需求说明的要求。分析的目的在于开发出高质量和具体的需求，这样你就能作出实用的项目估算并可以进行设计、构造和测试。

通常，把需求中的一部分用多种形式来描述，如同时用文本和图形来描述。分析这些不同的视图将揭示出一些更深的问题，这是单一视图无法提供的（ Davis 1995）。分析还包括与客户的交流以澄清某些易混淆的问题，并明确哪些需求更为重要。其目的是确保所有风险承

担者尽早地对项目达成共识并对将来的产品有个相同而清晰的认识。下面几章对需求分析中的任务进行了详细讨论：

- 第6章 —— 绘制系统关联图。
- 第9章 —— 建立数据字典。
- 第10章 —— 为需求建立模型。
- 第12章 —— 建立用户接口原型。
- 第13章 —— 确定需求优先级。

1) 绘制系统关联图　这种关联图是用于定义系统与系统外部实体间的界限和接口的简单模型。同时它也明确了通过接口的信息流和物质流。

2) 创建用户接口原型　当开发人员或用户不能确定需求时，开发一个用户接口原型 —— 一个可能的局部实现 —— 这样使得许多概念和可能发生的事更为直观明了。用户通过评价原型将使项目参与者能更好地相互理解所要解决的问题。注意要找出需求文档与原型之间所有的冲突之处。

3) 分析需求可行性　在允许的成本、性能要求下，分析每项需求实施的可行性，明确与每项需求实现相联系的风险，包括与其它需求的冲突，对外界因素的依赖和技术障碍。

4) 确定需求的优先级别　应用分析方法来确定使用实例、产品特性或单项需求实现的优先级别。以优先级为基础确定产品版本将包括哪些特性或哪类需求。当允许需求变更时，在特定的版本中加入每一项变更，并在那个版本计划中作出需要的变更。

5) 为需求建立模型　需求的图形分析模型是软件需求规格说明极好的补充说明。它们能提供不同的信息与关系以有助于找到不正确的、不一致的、遗漏的和冗余的需求。这样的模型包括数据流图、实体关系图、状态变换图、对话框图、对象类及交互作用图。

6) 创建数据字典　数据字典是对系统用到的所有数据项和结构的定义，以确保开发人员使用统一的数据定义。在需求阶段，数据字典至少应定义客户数据项以确保客户与开发小组是使用一致的定义和术语。分析和设计工具通常包括数据字典组件。

7) 使用质量功能调配　质量功能调配(QFD）是一种高级系统技术，它将产品特性、属性与对客户的重要性联系起来。该技术提供了一种分析方法以明确那些是客户最为关注的特性。QFD将需求分为三类：期望需求，即客户或许并未提及，但如若缺少会让他们感到不满意；普通需求；兴奋需求，即实现了会给客户带去惊喜，但若未实现也不会受到责备（ Zultner 1993;Pardee 1996）。

3.4 需求规格说明

无论你的需求从何而来，也不管你是怎样得到的，你都必须用一种统一的方式来将它们编写成可视文档。业务需求要写成项目视图和范围文档。用户需求要用一种标准使用实例模板编写成文档。而软件需求规格说明（requirement specification）则包含了软件的功能需求和非功能需求。你必须为每项需求明确建立标准的惯例，并确定在 SRS中采用任何惯例，以确保SRS的统一风格，同时读者也会明白怎样解释它。下列章节讨论了关于编写需求文档的几个方面：

- 第8章 —— 记录业务规范。
- 第9章 —— 采用SRS模板；为每项需求注上标号。
- 第18章 —— 指明需求来源；创建需求跟踪能力矩阵。

1) 采用SRS模板 在你的组织中要为编写软件需求文档定义一种标准模板。该模板为记录功能需求和各种其它与需求相关的重要信息提供了统一的结构。注意，其目的并非是创建一种全新的模板，而是采用一种已有的且可满足项目需要并适合项目特点的模板。许多组织一开始都采用IEEE标准830-1998(IEEE 1998)描述的SRS模板。要相信模板是很有用的，但有时要根据项目特点进行适当的改动。

2) 指明需求的来源 为了让所有项目风险承担者明白SRS中为何提供这些功能需求，要都能追溯每项需求的来源，这可能是一种使用实例或其它客户要求，也可能是某项更高层系统需求、业务规范、政府法规、标准或别的外部来源。

3) 为每项需求注上标号 制定一种惯例来为SRS中的每项需求提供一个独立的可识别的标号或记号。这种惯例应当很健全，允许增加、删除和修改。作了标号的需求使得需求能被跟踪，记录需求变更并为需求状态和变更活动建立度量。

4) 记录业务规范 业务规范是指关于产品的操作原则，比如谁能在什么情况下采取什么动作。将这些编写成SRS中的一个独立部分，或一独立的业务规范文档。某些业务规范将引出相应的功能需求；当然这些需求也应能追溯相应业务规范。

5) 创建需求跟踪能力矩阵 建立一个矩阵把每项需求与实现、测试它的设计和代码部分联系起来。这样的需求跟踪能力矩阵同时也把功能需求和高层的需求及其它相关需求联系起来了。在开发过程中建立这个矩阵，而不要等到最后才去补建。

3.5 需求验证

验证是为了确保需求说明准确、完整地表达必要的质量特点。当你阅读软件需求规格说明（SRS）时，可能觉得需求是对的，但实现时，却很可能会出现问题。当以需求说明为依据编写测试用例时，你可能会发现说明中的二义性。而所有这些都必须改善，因为需求说明要作为设计和最终系统验证的依据。客户的参与在需求验证（ requirement verification）中占有重要的位置，第14章还将进一步讨论它。

1) 审查需求文档 对需求文档进行正式审查是保证软件质量的很有效的方法。组织一个由不同代表（如分析人员，客户，设计人员，测试人员）组成的小组，对 SRS及相关模型进行仔细的检查。另外在需求开发期间所做的非正式评审也是有所裨益的。

2) 以需求为依据编写测试用例 根据用户需求所要求的产品特性写出黑盒功能测试用例。客户通过使用测试用例以确认是否达到了期望的要求。还要从测试用例追溯回功能需求以确保没有需求被疏忽，并且确保所有测试结果与测试用例相一致。同时，要使用测试用例来验证需求模型的正确性，如对话框图和原型等。

3) 编写用户手册 在需求开发早期即可起草一份用户手册，用它作为需求规格说明的参考并辅助需求分析。优秀的用户手册要用浅显易懂的语言描述出所有对用户可见的功能。而辅助需求如质量属性、性能需求及对用户不可见的功能则在 SRS中予以说明。

4) 确定合格的标准 让用户描述什么样的产品才算满足他们的要求和适合他们使用的。将合格的测试建立在使用情景描述或使用实例的基础之上（ Hsia, Kung, and Sell 1997）。

3.6 需求管理

当你完成需求说明之后，不可避免地还会遇到项目需求的变更。有效的变更管理需要对

变更带来的潜在影响及可能的成本费用进行评估。变更控制委员会与关键的项目风险承担者要进行协商，以确定哪些需求可以变更。同时，无论是在开发阶段还是在系统测试阶段，还应跟踪每项需求的状态。

建立起良好的配置管理方法是进行有效需求管理（requirement management）的先决条件。许多开发组织使用版本控制和其它管理配置技术来管理代码，所以你也可以采用这些方法来管理你的需求文档，需求管理的改进也是将全新的管理配置方法引入项目的组织中的一种方法。下列章节讨论了需求管理涉及到的各种技术：

- 第16章 —— 建立需求基准版本和需求控制版本文档。
- 第17章 —— 确定需求变更控制过程；建立变更控制委员会。
- 第18章 —— 进行需求变更影响分析；跟踪所有受需求变更影响的工作产品。
- 第19章 —— 使用需求管理工具。

1) 确定需求变更控制过程　确定一个选择、分析和决策需求变更的过程。所有的需求变更都需遵循此过程，商业化的问题跟踪工具都能支持变更控制过程。

2) 建立变更控制委员会　组织一个由项目风险承担者组成的小组作为变更控制委员会，由他们来确定进行哪些需求变更，此变更是否在项目范围内，估价它们，并对此评估作出决策以确定选择哪些，放弃哪些，并设置实现的优先顺序，制定目标版本。

3) 进行需求变更影响分析　应评估每项选择的需求变更，以确定它对项目计划安排和其它需求的影响。明确与变更相关的任务并评估完成这些任务需要的工作量。通过这些分析将有助于变更控制委员会作出更好的决策。

4) 跟踪所有受需求变更影响的工作产品　当进行某项需求变更时，参照需求跟踪能力矩阵找到相关的其它需求、设计模板、源代码和测试用例，这些相关部分可能也需要修改。这样能减少因疏忽而不得不变更产品的机会，这种变更在变更需求的情况下是必须进行的。

5) 建立需求基准版本和需求控制版本文档　确定一个需求基准，这是一致性需求在特定时刻的快照。之后的需求变更就遵循变更控制过程即可。每个版本的需求规格说明都必须是独立说明，以避免将底稿和基准或新旧版本相混淆。最好的办法是使用合适的配置管理工具在版本控制下为需求文档定位。

6) 维护需求变更的历史记录　记录变更需求文档版本的日期以及所做的变更、原因，还包括由谁负责更新和更新的新版本号等。版本控制工具能自动完成这些任务。

7) 跟踪每项需求的状态　建立一个数据库，其中每一条记录保存一项功能需求。保存每项功能需求的重要属性，它包括状态（如已推荐的，已通过的，已实施的，或已验证的），这样在任何时候都能得到每个状态类的需求数量。

8) 衡量需求稳定性　记录基准需求的数量和每周或每月的变更（添加、修改、删除）数量。过多的需求变更“是一个报警信号”，意味着问题并未真正弄清楚，项目范围并未很好地确定下来或是政策变化较大。

9) 使用需求管理工具　商业化的需求管理工具能帮助你在数据库中存储不同类型的需求，为每项需求确定属性，可跟踪其状态，并在需求与其它软件开发工作产品间建立跟踪能力联系链。

3.7 项目管理

软件工程管理方法在本质上与项目的需求过程是紧密相关的。项目计划建立在功能基础

之上，而需求变更会影响这些计划。因此，项目计划应能允许一定程度的需求变更或项目范围的扩展。如果刚开始，需求不能确定，你可以选择一种软件开发方法生存周期以允许这种不确定性，并在弄清后逐渐实施。关于需求工程项目管理（ project management）方法的更详细内容将在以下章节讨论：

- 第5章 —— 编写文档和管理与需求相关的风险。
- 第15章 —— 基于需求的项目计划。
- 第16章 —— 记录需求开发和管理中的工作。

1) 选择一种合适的软件开发方法生存周期　经典的瀑布法软件开发生存周期只适用于需求说明在项目早期即可全部完成的情况。你的组织应根据不同类型的项目和需求说明的不同程度选择几种不同的方法（McConnell 1996）。如果需求说明在项目早期无法全部确定，则从最为清晰易懂的需求开始，建立一个健壮的可修改的结构，再逐渐增加补充。实现了部分特性的产品可作为早期版本发布（Gilb 1998）。

2) 基于需求的项目计划　随着需求细节不断变得清晰、完善，项目开发计划的进度安排将会不断改变。一开始可以根据项目视图和范围对开发功能需求所需的工作量作一估算，建立在不甚完善的需求基础之上的成本、进度安排的估计很不可靠，但随着需求说明的完善，估计也会得到不断改善。

3) 发生需求变更时协商项目约定　当在项目中添加新的需求时，估计一下你是否能在目前安排下，利用现有资源保质保量完成。如果不能，将项目的实现与管理联系起来，协商一下新的、切实可行的约定（Humphrey 1997）。如果协商不成功，则将可能的后果和更新项目风险管理计划联系起来，以反映出对项目成功的新的不利因素。

4) 编写文档和管理与需求相关的风险　采用自由讨论的方法并将与需求相关的项目风险编写成文档。利用各种方法来减轻或阻止这些风险，实施这些方法并跟踪其发展及效果。

5) 跟踪需求工程所耗的工作量　记录需求开发和管理活动所花费的工作量。利用这些数据可以评估计划的需求活动是否已达到所期望的要求，并可以为将来项目的需求工程提供更好的所需资源的计划。

下一步：

- 回到第1章的下一步中你已明确了与需求有关的一些问题。从本章中学到的推荐的实践方法，可能会有助于解决这些问题。针对建议的每一种方法，要在你组织中发现那些可能给其实现带来困难的障碍。
- 将在先前步骤中被确认是好的需求方法整理成一张表。针对每个方法，指明你的项目的能力水平：专家，熟练者，生手或新手。如果你的队伍中无一人熟练的话，就得要求项目的某个参与者学习更多的知识，并将他所学的教给队伍中的其他人。