通过用例应用需求管理

Roger Oberg、Leslee Probasco

和 Maria Ericsson

Rational Software 白皮书

TP505 （版本 1.4）

Rational®

the software development company

目录

如果您对需求管理一无所知或者一知半解，但有志于改进需求过程，那么本文将为您提供一个框架，您可以利用它开发自己的方案。

用例和软件需求规约(SRS)

为了让读者更好地理解需求管理工作流，作者从 Rational Software 的 Unified Process 以及工业标准统一建模语言特地挑选了建模语言特地挑选了和需求管理工件予以说明。Unified Process 和统一建模语言都推荐使用一种用例驱动的软件工程流程。

因此，本文描述了一种用于指定软件需求的用例驱动方案。这些工作流还可代替用例模型和用例，或作为它们的补充,与更传统的软件需求规约（如 IEEE 标准）一起使用。

流程时代的软件和系统开发

对大多数软件和系统开发团队来说，与过去自由的日子相比，20 世纪 90 年代是一个强调流程的时代。评测和验证有效的软件开发流程的标准得到推广和普及。许多论述软件开发流程的书籍和文献以及关于业务建模和重建的的相关材料纷纷出版。不断涌现出的软件工具已经帮助人们制定和应用有效的软件开发流程。在这十年内，全球经济对软件的依赖程度加深，它推动着开发流程的发展，提高了系统质量。

既然如此，那么今天频频发生的软件项目失败的事件又如何解释呢？即使不是大多数，但为什么仍有那么多的项目受到延期、预算超支和质量问题的困扰呢？随着我们的业务、国家经济和日常活动越来越依赖于系统，如何才能提高系统的质量？

像往常一样，答案就在于该行业的人员、工具和流程。需求管理通常在软件开发出现问题时作为解决方案提出来，但对于这门学科的改进则较少关注。

本文介绍有效需求管理流程的元素，特别强调成功实施需求管理所面临的障碍。

需求管理除了应用于纯软件项目以外，同样也应用于软件只占最终结果的一部分或根本不包括在内的项目。为方便起见，本文以后将使用“系统”这个词来指代任何或所有这些项目。然而，正是软件开发的抽象本质本身，或者和硬件一起，使需求管理复杂化了，因此本文的重点在于软件开发。

为什么要管理需求？

简单地说，系统开发团队之所以管理需求，是因为他们想让项目获得成功。满足项目需求即为成功打下了基础。若无法管理需求，达到目标的几率就会降低。

以下最近收集的证据很有说服力：

- Standish Group 从 1994 年到 1997 年的 CHAOS Reports 证实,导致项目失败的最重要的原因与需求有关。[1]
- 1997 年 12 月，Computer Industry Daily 报导了 Sequent Computer Systems 公司的一项研究，该公司对美国和英国 500 名 IT 经理作调查后发现，百分之七十六的受访者在他们的事业中经历过完全的项目失败。其中提到最多的导致项目失败的原因就是“变更用户需求”。[2]

为什么要管理需求？避免失败就是一个很充分的理由。提高项目的成功率和需求管理所带来的其他好处同样也是理由。Standish Group 的 CHAOS 报告进一步证实了与成功项目关系最大的因素是良好的需求管理。

什么是需求？

理解需求管理的第一步就是对什么是需求管理达成共识。Rational 把需求定义为“（正在构建的）系统必须符合的条件或具备的功能”。电气和电子工程师学会使用的定义与此类似。

著名的需求工程设计师 Merlin Dorfman 和 Richard H. Thayer 提出了一个包容且更为精练的定义，它特指软件方面但不仅仅限于软件：

“软件需求可定义为：

- 用户解决某一问题或达到某一目标所需的软件功能。
- 系统或系统组件为了满足合同、规约、标准或其他正式实行的文档而必须满足或具备的软件功能。”[3]

什么是需求管理？

由于需求是正在构建的系统必须符合的事务，而且符合某些需求决定了项目的成功或失败，因此找出需求是什么，将它们记下来，进行组织，并在发生变化时对它们进行追踪，这些活动都是有意义的。

换句话说，需求管理就是：

- 一种获取、组织并记录系统需求的系统化方案，以及
- 一个使客户与项目团队对不断变更的系统需求达成并保持一致的过程。

这个定义与 Dorfman 与 Thayer 以及 IEEE 的软件需求工程”的定义相似。需求工程包括获取、分析、规定、验证和管理软件需求，而“软件需求管理”则是对所有相关活动的规划和控制[4]。这里介绍的以及 Rational Software 提出的需求管理定义包括了所有这些活动。它们的区别主要在于这里选用了“管理”这个词，而不是“工程”。管理这个词更合适用来描述所有涉及到的活动，并且它准确地强调了追踪变更以保持项目干系人与项目团队之间共识的重要性。

对那些不熟悉“引出”这个词的人来说，它可定义为团队用来获取或发现项目干系人请求，确定请求后隐藏的真正需要，以及为满足这些需要对系统提出的一组适当需求。

需求管理问题

一个目的在于确保系统符合人们对其期望的流程面临着哪些困难呢？当它真正在实际项目实施时，困难就暴露出来了。图 1 显示了 1996 年对开发人员、经理和质量保证人员所做的一次调查结果。该图显示了经历过最常提到的需求相关难题的受访者比例。

图 1- 常见的需求问题

下面列出了更多与需求有关的问题：

- 需求不总是显而易见的，而且它可来自各个方面。
- 需求并不总是容易用文字明白无误地表达。
- 存在不同种类的需求，其详细程度各不相同。
- 如果不加以控制，需求的数量将难以管理。
- 需求相互之间以及与流程的其他可交付件之间以多种方式相关联。
- 需求有唯一的特征或特征值。例如，它们既非同等重要，处理的难度也不同。
- 需求涉及众多相关利益责任方，这意味着需求要由跨职能的各组人员来管理。
- 需求发生变更。
- 需求可能对时间敏感。

当这些问题与需求管理和处理技能不足以及缺乏易用工具等情况一同出现时，许多团队都对管理好需求不抱希望了。Rational Software 已经开发出指导团队提高需求管理技能和流程的专业技术。此外，Rational Requisite®Pro 是一个可获得的、有效进行自动化管理需求的工具。

需求管理技巧

为了解决上述问题，Rational 鼓励对关键技巧的开发。下面将要介绍的这些技巧是按顺序给出的，但在有效的需求管理流程中，它们以不同的顺序连续应用。在此，它们将以新项目在第一次迭代时可能使用的顺序出现。

图 2- 问题分析步骤

关键技巧 1：分析问题

进行问题分析是为了理解业务问题，确定项目干系人的最初需要，并提出高层解决方案。这些推理和分析行为找出“隐藏在问题背后的问题”。

在问题分析过程中，将对实际问题的说明取得一致，并确定有关的项目干系人。初始解决方案的界限和约束从技术和业务两个方面来定义。在适当的时候，项目的商业理由还分析期望从系统获得的投资回报。

关键技巧 2：理解项目干系人需要

需求有许多来源。它们可能来自于对项目结果感兴趣的任何人。客户、合作伙伴、最终用户以及领域专家都是某些需求的来源。而管理人员、项目团队成员、业务策略和管理机构是另外一些需求源。

因此，掌握如何确定哪些人员应该是需求源、如何获得这些需求源以及如何从中获取信息是很重要的。作为提供这些信息主要来源的个人在本项目中称为“项目干系人”。

如果您正在开发一个在您公司内部使用的信息系统，那么在开发团队中应包括具有最终用户经验和业务领域专业知识的人员。讨论一般从业务模型一级开始，而不从系统一级开始。如果正在开发一个要在市场上出售的产品，那么您可以充分调动营销人员，以便更好地了解该市场中用户的需要。

获取需求的手段包括访谈、集体讨论、概念原型设计、问卷调查和竞争性分析等。需求获取的结果可能是一份图文并茂的请求或需要列表，并按相互之间的优先级列出。

关键技巧 3：定义系统

定义系统指的是解释项目干系人需求，并整理为对要构建系统的意义明确的说明。在系统定义初期，需要决定需求的构成、文档格式、语言规范、需求等级、请求优先级和预计工作量、技术和管理风险以及系统规模等。系统定义活动还可包括与最关键的项目干系人请求直接联系的初期原型和设计模型。

我们使用了“说明”这个词而没有用“文档”，是为了避免在后者常见的使用中发现的固有缺陷。说明可以是书面文档、电子文件、图像或用于表达除系统本身以外的系统需求的任何表示方式。

系统定义的结果是用自然语言和图解方式表达的系统说明。后面几节将介绍推荐使用的一些说明格式。

原则 55：在编写比较正式的模型前，先使用自然语言进行编写

在第一次编写正式模型时，往往要用自然语言描述模型，而不是直接得到解决方案系统。请考虑以下示例：

> 要打长途电话时，用户应该拿起电话。系统将回应拨号音。用户拨打号码“9”，系统回应一种特殊的拨号音…
>
> 系统有四种状态：空闲、拨号音、特殊拨号音和连接状态。要使系统从空闲状态转向拨号音状态，请拿起电话。要从拨号音状态转到特殊拨号音状态，请拨号码“9”。

注意在后一个例子中，文字未起任何帮助读者理解的作用。

- Alan M. Davis, 201 Principles of Software Development, 1995

关键技巧 4：管理系统规模

项目规模由分配给它的需求集定义。管理项目规模，使之适合可用资源（时间、人力和资金），是成功管理项目的关键。管理规模是一个持续不断的活动，需要迭代式和递增式开发，这就将项目细分为更易于管理的若干个小部分。

使用需求属性（如优先级、工作量和风险）作为协商应包含需求的基础，对管理规模而言是相当有用的技巧。侧重于属性，而不是需求自身，将减少通常对敏感问题的争论。

这也有助于培养小组负责人的谈判技巧，还有助于在项目中为组织培养推介人，对于客户而言也是如此。产品/项目推介人应能够代表组织拒绝那些超出可用资源的规模变更，也应有相应能力扩展资源以适应扩大的规模。

关键技巧 5：改进系统定义

对高层系统定义达成一致并充分理解了系统初始规模后，投入资源改进系统定义不仅有可能，而且也是经济的。改进系统定义包括两个重要的考虑事项：制定更详细的高层系统说明和验证系统是否符合项目干系人需要，是否按说明运行。

说明通常是项目团队的重要参考材料。在制定说明时一定要考虑受众。人们常会犯一个错误，那就是用复杂定义表示构建起来很复杂的东西，尤其是在受众无法或不愿意耗费精力去思考某些重要的需要取得一致认识时候。这就难以向项目团队内外的人员解释系统目的。相反，你可能会发现需要为不同的受众编制不同类型的说明。本文介绍了详细自然语言、正式文字和图解说明推荐使用的格式。确定说明格式后，改进将持续贯穿整个项目。

关键技巧 6：管理变更的需求

不管您多么认真地定义需求，需求终将改变。实际上，一些变更是非常值得的！这意味着您的团队需要与项目干系人保持密切联系。能否适应变更需求是评测团队的项目干系人敏感度和运作灵活性的一个尺度 - 敏感度和灵活性都是对项目成功有贡献的团队特征。变更不是敌人，而没有管理的变更才是真正的敌人。

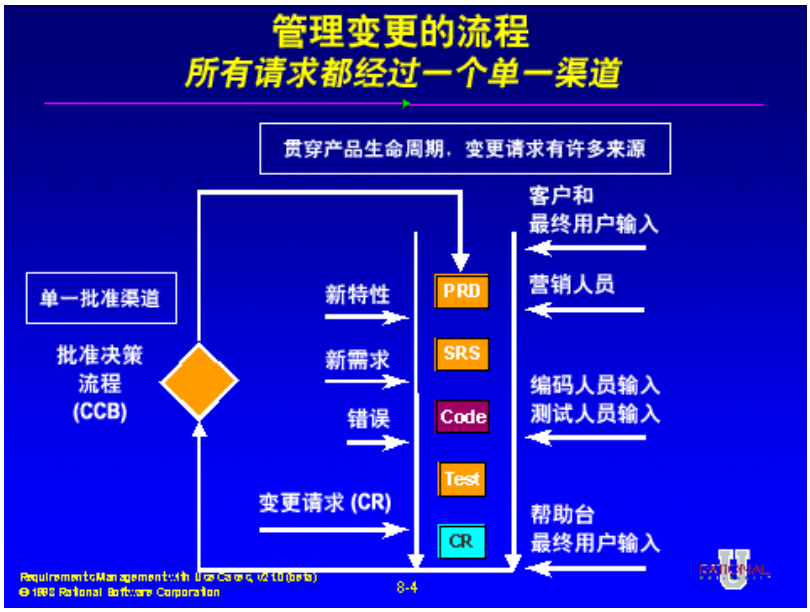

图 3 - 变更管理流程

需求变更表明多少需要耗费一些时间来实施某个特定的功能，而且对一个需求的变更对其他需求可能有影响。管理需求变更包括这样一些活动：设立基线、追踪每个需求的历史、确定哪些依赖关系值得追踪、在相关项之间建立可追踪关系以及维护版本控制等。如图 3 所示，建立变更控制或批准流程也很重要，它要求由指定的团队成员来复审所有提议的变更。有时候这一单一的变更控制渠道称为变更控制委员会(CCB)。

重要的需求概念

为了在项目中运用需求管理技巧，参与项目的每个人理解某些基本的需求管理概念是很有裨益的。这些功能包括：

- 需求类型
- 功能交叉的团队
- 可追踪性
- 多维属性
- 变更历史

需求类型

系统越大越复杂，出现的需求类型就越多。一个需求类型不过是指需求的一个类。通过确定需求类型，团队可以把大量需求组织成意义明确且更容易管理的组。在一个项目中建立不同类型的需求有助于团队成员对变更请求进行分类，并使相互之间的沟通更为清楚明确。

通常，一类需求可以细分即分解成其他类型的需求。业务规则和前景声明包括高层次的需求，团队可以从中导出用户需要、特性和产品需求类型。用例和其他建模形式驱动设计需求，该需求可分解为软件需求，并可以用分析设计模型来说明。测试需求源于软件需求，它被分解为具体的测试过程。如果既定项目中有成百上千个，甚至上万个需求实例时，对需求进行分类可以使项目更容易管理。

功能交叉的团队

与诸如测试或应用程序建模等流程不同（这些流程可在单个业务组中进行管理），需求管理涉及了每一个能够为开发流程提供专门技术的个人。其中应包括那些代表客户和业务预期的人。开发经理、产品经理、分析员、系统工程师甚至客户都应该参与进来。需求团队还应包括创建系统解决方案的人 - 工程师、构架设计师、设计员、程序员、技术文档编写员以及其他提供技术支持的个人。测试员和其他质量保证人员应当作重要的团队成员。

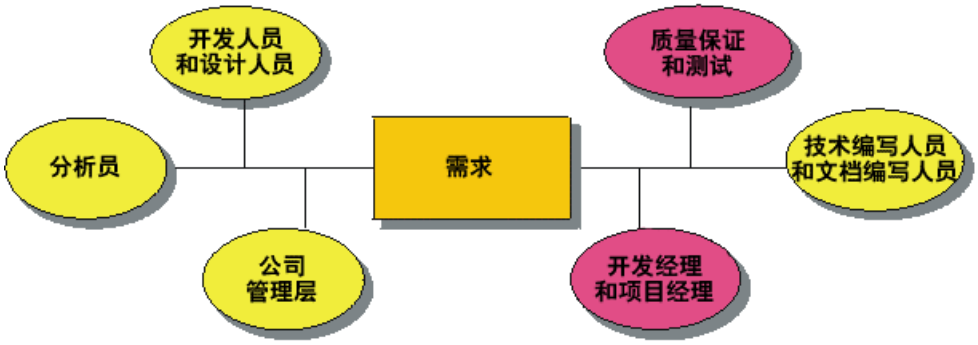

图 4- 功能交叉的需求团队

通常，创造和维护需求类型的职责可按照功能范围来分配，从而进一步优化大型项目的管理。需求管理的功能交叉性质是这门学科非常具有挑战性的一个方面。

可追踪性

如需求类型说明所述，没有一个需求表述是孤立存在的。项目干系人请求与提议用于满足这些请求的产品特性有关。产品特性与指定特性的功能性和非功能性行为的各个需求相关。测试案例与它们检验和验证的需求相关。有一些需求可能依赖于其他需求，也可能相互排斥。团队为了确定变更带来的影响，保证系统符合预期，就必须理解、记录并维护这些可追踪性关系。尽管可追踪性是需求管理中最难实施的概念之一，但它是适应变更所必不可少的。建立明确的需求类型，吸收功能交叉人员的参与，可使可追踪性更容易实施和维护。要了解需求可追踪性策略的更多信息，请参见白皮书 “通过用例进行需求管理的可追踪性策略”[5]。

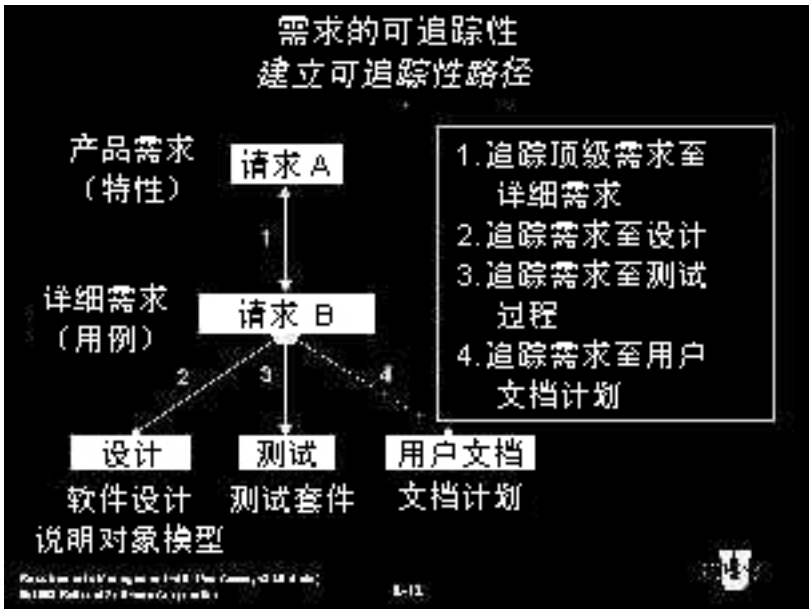

图 5 - 需求可追踪性

多维属性

每一个需求类型都有属性，每一个独立需求都有不同的属性值。例如，可为需求分配优先级，确定其来源和原理，委派给某个职能范围内的特定子团队进行管理，指定一定的难度，或者与系统的某个迭代关联关系起来。图 6 对此作了说明，该图显示了 Rational RequisitePro 需求管理工具中 Learning Project 的一个特性需求类型的属性。正如屏幕的标题所暗示，需求类型和每种类型的属性是为整个项目定义的，由此确保了团队上下使用的一致性。

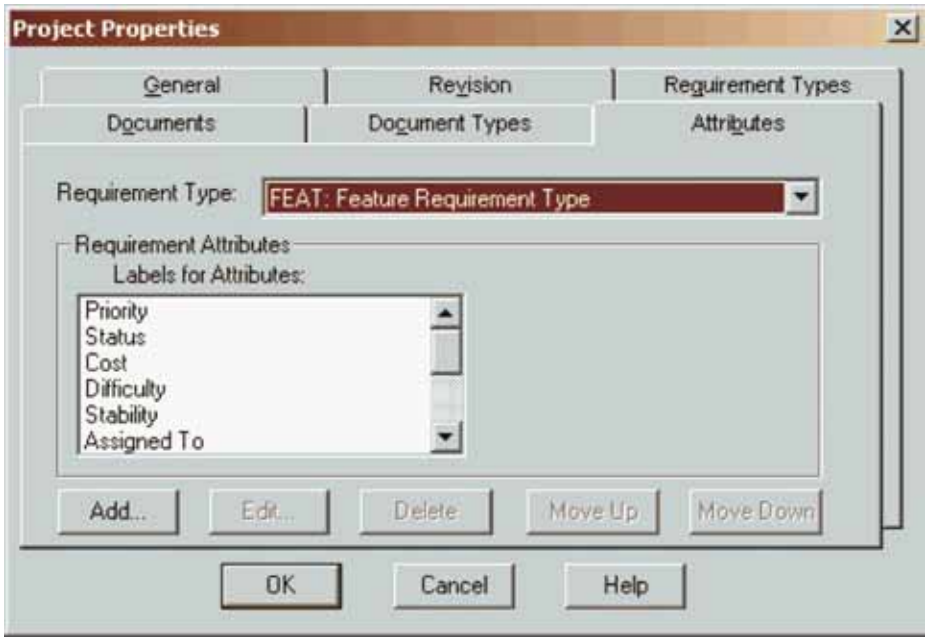

图 6 - 定义特性的需求属性

图 7 显示了 RequisitePro 中某个项目的特性需求实例。注意，即使未完整地显示每个需求，但我们从属性值即可了解每个需求的很多内容。在这个例子中，需求的优先级和难度 - 毫无疑问是由团队的不同成员指定的 - 有助于团队兼顾考虑项目干系人优先级，以及对难度属性值所反映工作量的大致估计，把项目规模限制在可用资源和时间的合理范围内。在更详细的需求类型中，优先级工作量属性可能有更多的具体值（如预计时间、代码行等），从而进一步改进规模。需求的多维性以及不同类型的需求（每种类型都有自己的属性）对于组织大量需求和管理项目的整体规模来说是不可或缺的。

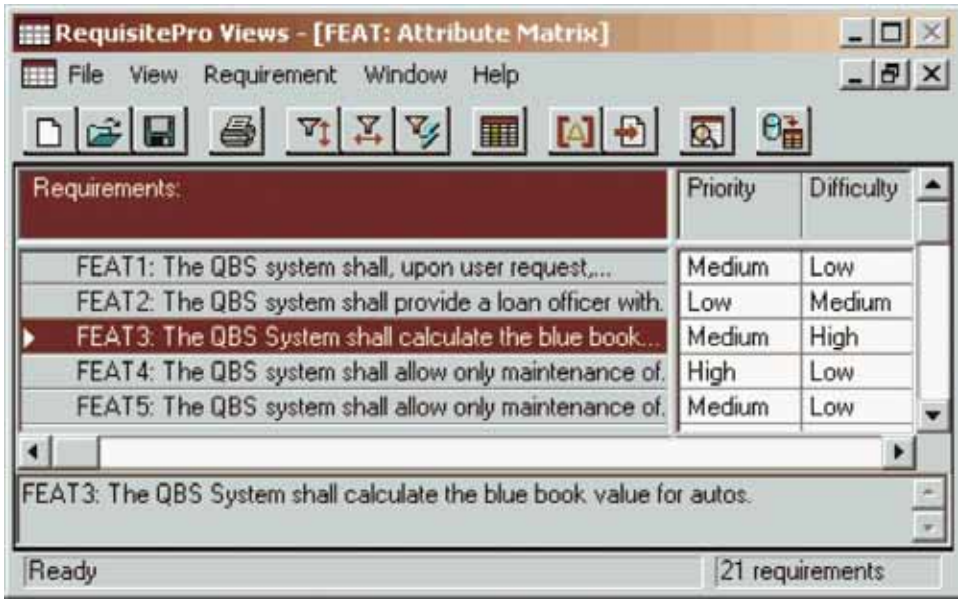

图 7 - 设置各个需求的属性值

变更历史

单个需求和需求集合都有历史记录，并且内容随着时间的推移而更具有含义。变更在所难免，而且变更有助于与环境变化和技术发展保持同步。记录项目需求版本有助于团队领导人找出变更项目的原因，如新的系统发布等。需求集合可能与某一个软件版本相关，理解这一点有助于人们扩大对变更的管理，降低风险，提高降低风险，提高实现重大里程碑的几率。随着单个需求发展，理解它们的变更历史是很重要的：变更内容、起因、时间和谁授权变更等。

将需求管理付诸实践

需求管理采用上面介绍的关键技巧和概念成功地识别并解决问题。

为了建立一个真正满足客户需求的系统，项目团队首先必须确定系统要解决的问题。然后，团队必须确定项目干系人，从中获得业务和用户需要，对其进行描述，并区分它们的优先级。从这一组高层期望或需求出发，对产品或系统特性集达成一致意见。

详细的软件需求应该以客户和开发团队都能理解的形式写下。我们发现，使用客户的语言来描述这些软件需求在取得理解和共识的过程中是最有效的。这些详细的软件需求随后用作系统设计规约的输入，或者作为实施和验证所需的测试计划及过程的输入。软件需求还应该促进初始用户文档规划及设计。

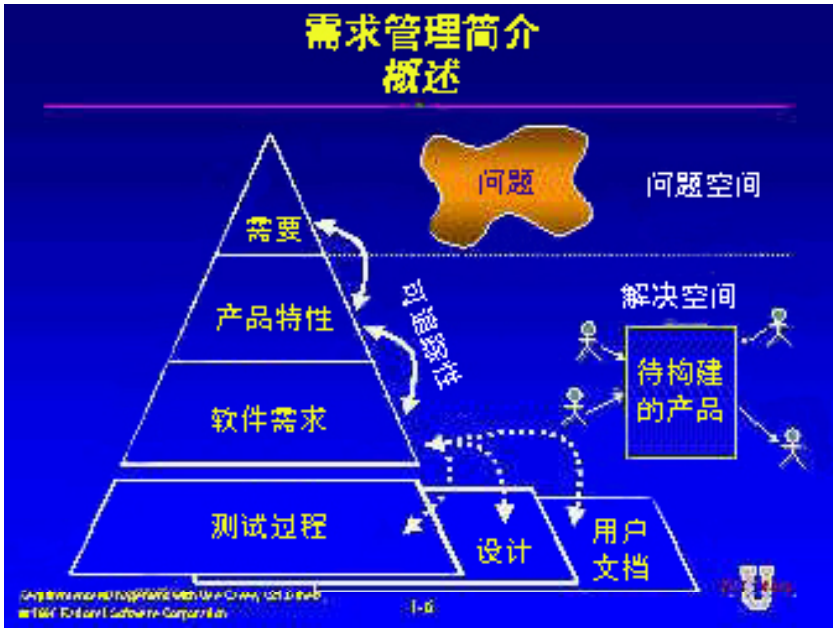

图 8 - 需求管理结构概述

为了推动需求管理，项目团队应该：

- 对项目的常用词汇表取得一致。
- 制定系统前景，描述系统将要解决的问题以及系统的主要特性。

- 至少获得五个重要领域的项目干系人需要：功能、可用性、可靠性、性能和可支持性。
- 决定使用哪些需求类型。
- 为每一个需求类型选择属性及属性值。
- 选择需求的说明格式。
- 确定创造一种或多种需求类型的团队成员，以及那些对此有贡献或仅仅进行查看的团队成员。
- 决定所需的可追踪性。
- 制定变更需求需遵守的提议、复审、决议程序。
- 开发一个追踪需求历史的机制。
- 为团队成员和管理人员创建进度和状态报告。

这些必要的需求管理活动与行业、开发方法和需求工具无关。它们同时也很灵活，能够在最严格、变化速度最快的应用软件开发环境下执行有效的需求管理。

文档简介

用文档来描述需求的决定值得认真考虑。一方面，书面记录是人们广泛接受的一种交流形式，对大部分人来说，这是自然不过的了。另一方面，项目的目标在于产生系统，而不是文档。

常识和经验告诉我们，问题不在于是不是要记录需求，而是怎么记录。文档模板为需求管理提供了一种统一的格式。Rational RequisitePro 不仅提供这些模板，还提供其他特性，将文档内的需求链接到一个包含所有项目需求的数据库。这种独一无二的特性不仅允许需求自然地记录下来，同时还能存放在关系型数据库里，便于访问和管理。

需求管理：工作流明细

根据领域的不同，需求管理可遵循的方案有无限多种。面的方案给出了六个详细的工作流，它们适用于每一个关键的需求管理技巧，但也可以应用到任何领域。

下面的工作流图摘自 Rational Unified Process [6]需求工作流明细。这些工作流用角色、活动和工件（输入或输出）表示，图 9 的活动图对它们进行了概括。旁边的文字简单描述了每个工作流，希望藉此增进读者对改进需求管理流程的理解和兴趣。关于 Rational Unified Process 的更多信息，可参考 www.rational.com。

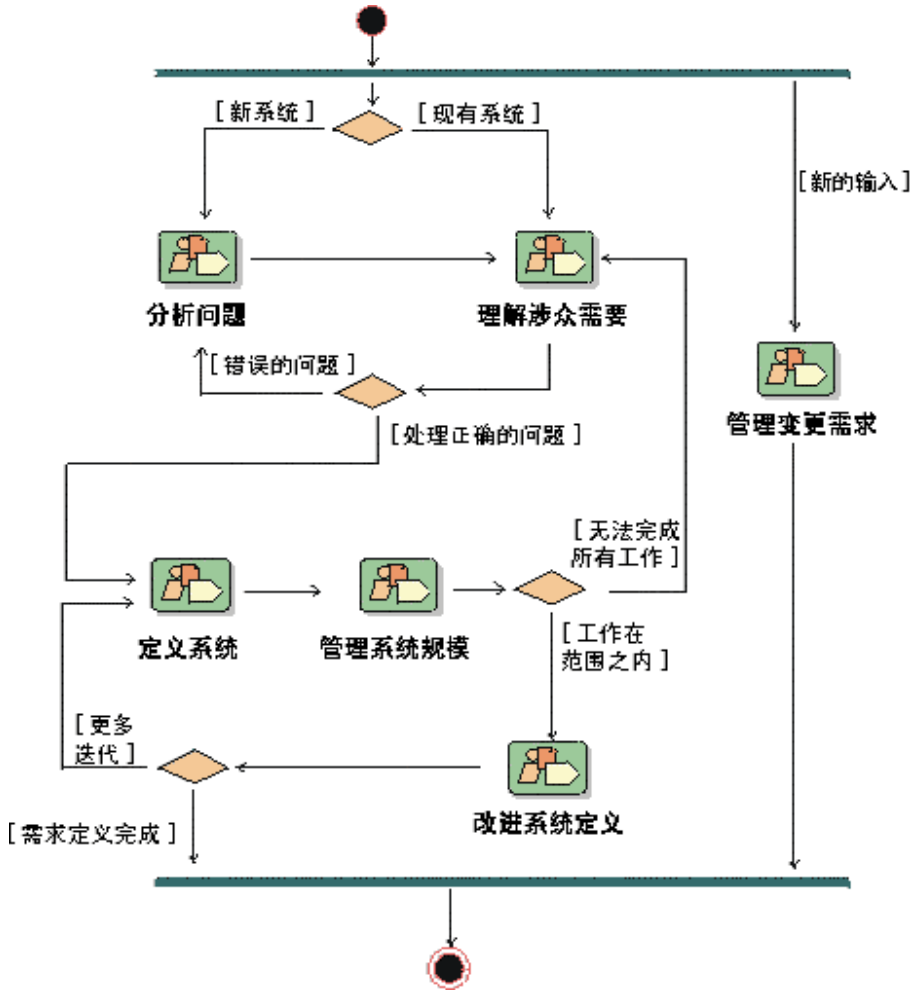

图 9 - Rational Unified Process **中的需求工作流**

工作流明细：分析问题

在问题分析中，主要的活动是制定项目前景。此活动的结果是产生了一个前景文档，它确定了待建系统的高级用户或客户视图。前景文档将初始需求作为关键特性表述，这些特性是系统为了解决重大问题并满足关键项目干系人需要而必须具备的。系统分析员在此工作流中扮演主要角色。系统分析员应该具有问题分析领域的专业知识，对问题有一定的理解，还应该能描述其认为可以解决问题的流程。此阶段要求各个项目项目干系人积极参与，还应该考虑所有相关的项目干系人请求。

要开始管理依赖关系活动，应该为职责分配属性，如基本原理、相对值或优先级以及请求的来源等。随着前景的发展，分析员确定可能用例的用户和系统（主角）。主角是用例模型的首要要素，它们将定义系统的功能性和非功能性技术需求。

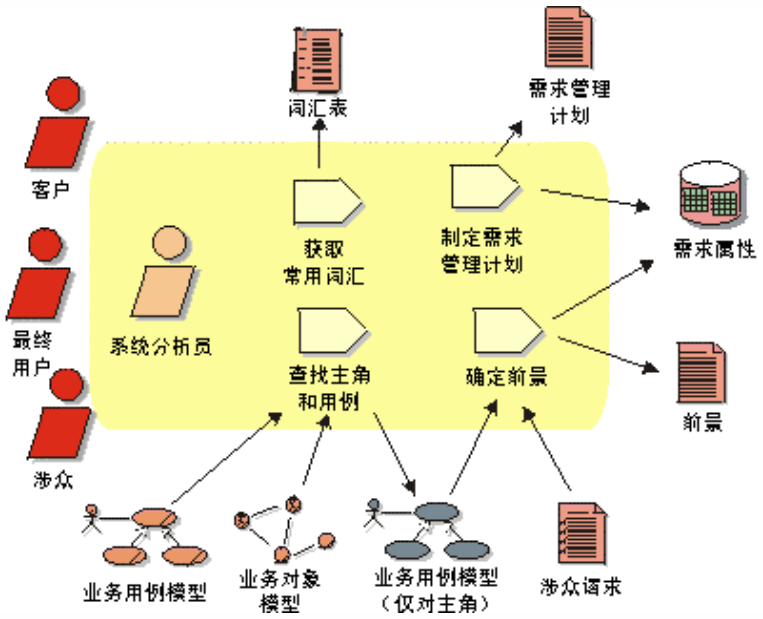

图 10 - 分析问题

工作流明细：分析问题

启动：一个或多个认识到问题存在的项目干系人启动工作流。

开发团队中的系统分析员可以和这几个最初的项目干系人展开会话，帮助他们描述需要解决的问题。对所认识问题的简要说明达成一致意见是很重要的。下表列出了**问题说明**的关键元素：

问题	确定问题
对项目干系人影响	列出受问题影响的项目干系人
问题的影响	描述问题的影响
成功的解决方案	列出成功解决方案的一些主要优点

问题说明简明扼要解释了项目的目的。问题分析员深入调查所有项目干系人请求和初始商业理由，包括显著优点以及大致估计的成本等。在定义问题说明的同时，团队还应该编写词汇表，记录常用术语并统一其定义。

用例模型简介

用例模型包括主角、用例以及它们之间的关系。主角代表了必须与系统交换信息的所有事物，包括通常所谓的用户。当主角使用系统的时候，系统执行一个用例。好的用例是一个事务序列，该序列为主角提供一个可评测的价值结果。用例集合是系统的完整功能。

—Jacobson I., Christerson M., Jonsson P., Overgaard G., Object-Oriented Software A Use Case Driven Approach, Addison Wesley - ACM Press, 1992

问题分析员还确定系统的主要主角。主角是系统的用户或者将与之交换信息的其他任何系统。在这一阶段，问题分析员应该简单确定主角与系统交互的一些显而易见的方式。说明应面向业务流程而非系统行为。例如，预算程序可能会允许一个类型的主角“制定部门预算”，而其他类型的主角可以“合并部门预算”。系统分析员以后可以将它们用到其他用例中，这些用例与特定系统行为结合起来更有意义。例如，“制定部门预算”可以产生像“导入电子表格信息”以及“创建预算视图”等系统 spreadsheet 用例。

上述问题分析会话通常进行不止一次，会话对象可能是不同的项目干系人，并且中间还夹带开发团队的内部会话。与项目干系人打交道的系统分析员将主持一次与开发团队成员的会话，展望问题的技术解决方案，从初始项目干系人输入中导出特性，并起草前景说明，即待建系统的第一个定义。为了方便初始项目干系人理解提议的解决方案，系统分析员可以利用建模工具或手工绘图技术来完善前景说明。

要从多方面向初始项目干系人咨询，以帮助改进问题说明，限制可能解决方案的个数和规模。项目干系人和系统分析员就关键特性的优先级进行谈判，对资源及开发资源所需的工作量取得一般性的理解，藉此来管理该工作流中的依赖关系。尽管优先级和工作量/资源的估计不可避免会发生变化，但提早管理依赖关系可建立起一个在整个开发生命周期中一直延续使用的重要模式。这是规模管理的本质所在，是一个项目成功的早期预报器。

在经过几次草案更新后，前景文档进入团队必须决定是否对额外需求获取进行投资的阶段。同时，商业理由的批准过程也分别开始启动了。本文不打算深入探讨商业理由，只对其进行简单说明。商业理由描述：

- 环境（产品领域、市场和规模），
- 技术方案，
- 管理方案（时间安排、风险、对成功的客观评测方法），
- 以及财政预测。

工作流明细：理解项目干系人需要

如果初始问题分析证明增加投资是合理的，则郑重开始理解项目干系人需要工作流。这一阶段的关键活动是获取项目干系人请求。主要结果是收集确定优先级的项目干系人需要和特性，利用此结果可改进前景文档，加深对需求属性的理解。另外，在这个工作流中，您可以根据用例和主角对系统展开讨论。另一个重要输出结果是经过更新的词汇表，它使团队成员对常用词汇有一致的理解。

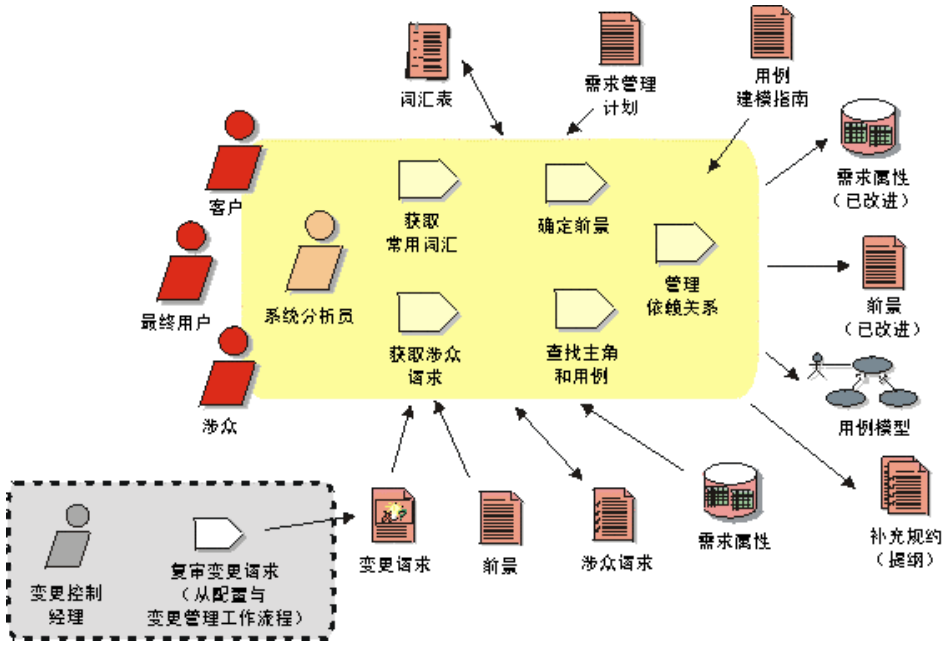

图 11 - 理解项目干系人需要

工作流明细：理解项目干系人需要

系统分析员和关键项目干系人通过访谈、专题讨论会、示意板、业务流程用例和其他手段，确定更多的项目干系人，获取他们的请求，确定关键需要和特性。这些会话由一个或多个系统分析员主持进行。需求专题讨论会是最有用的需求获取手段。该流程包括用户、帮助台人员、企业主、测试员以及其他对提议项目的结果有利害关系的个人。项目干系人请求通常是不明确、重叠甚至是离谱的。除正式的获得结果外，项目干系人请求还可以用格式设计很好的文档、数据库的缺陷和扩展请求或者电子邮件和群件线程等形式表达。系统分析员记录已确定的关键项目干系人需要，对其进行分类和区分优先次序。

理解项目干系人需要："取悦客户"从何处开始

项目干系人请求要尽可能在请求项目干系人使用的语言和格式中获取。与后继的需求类型（通常由具有丰富流程知识和技术的项目团队成员创造）不同，项目干系人请求通常很难表达清楚。项目干系人请求经常交叉或重复。而且项目干系人请求的表达形式多种多样，从纸条到扩展请求数据库等等都有。

分析员（或代表分析员角色的团队）必须复审所有需求，对其进行解释、分类，甚至还要重新录入（不重写），在前景文档中将其翻译成关键项目干系人需要及系统特性。根据开发的严格程度以及工具的可用性，可应用一部分或所有项目干系人请求、需要与特性之间的可追踪性，帮助项目干系人理解在决定系统需求时如何解释清楚他们的请求和需要。

通过应用理解项目干系人需要工作流，说明获取请求和满足项目干系人需要的非常重要事项，对建立项目干系人对开发团队能力的信心而言具有重大意义。

对项目干系人需要有了更好的理解之后，开发团队里的系统分析员改进前景文档，将主要精力放在制定"产品定位说明"上。该说明用简洁的两三句话确立项目的显著价值。说明应包括预期用户、项目解决的问题、

所带来的利益，以及它所取代的竞争者。所有的团队成员都应该理解该项目的主题。系统分析员还更新词汇表，促进团队对术语的共同理解。

从多方面向关键项目干系人咨询，以便对从理解项目干系人需要得到的新特性的优先级进行协商，获得对当前开发新特性所需资源和工作量的理解。利用问题分析，在该工作流中管理依赖关系有助于管理项目的规模。它还建立起项目干系人请求、需要与系统特性之间的可追踪性，让项目干系人相信他们的建议确实得到考虑。

工作流明细：定义系统

最初的两个需求工作流明细（分析问题和理解项目干系人需要）创建了关键系统定义的早期迭代（包括前景文档指定的特性）、用例模型的第一个大纲，以及初始需求属性。下一个工作流明细即定义系统，通过改进特性定义，添加新主角、用例和补充规约，完善了高级系统需求说明。

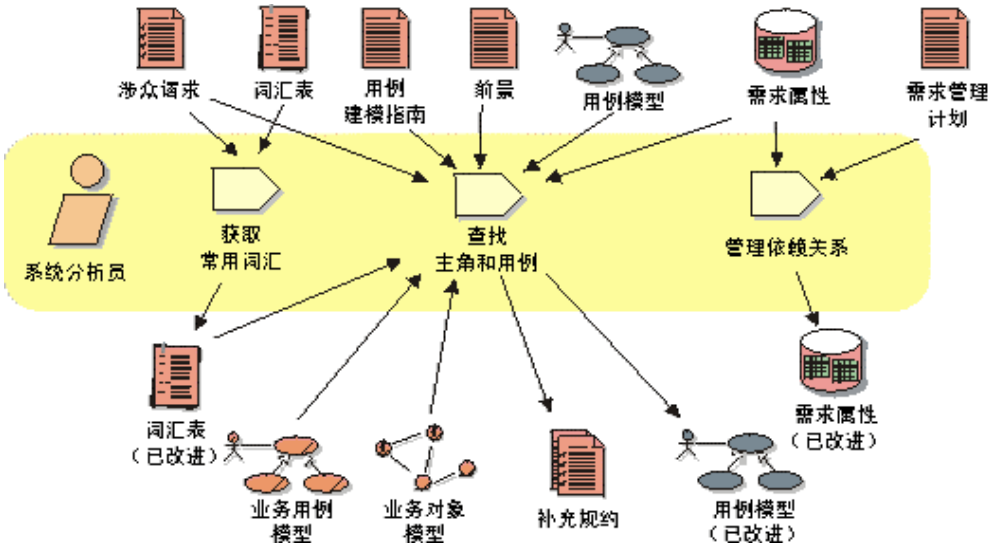

图 12 - 定义系统

工作流明细：定义系统

更新词汇表是为了反映当前对术语的理解，这些术语用于描述在用例模型及补充规约中捕获的特性和需求。

系统分析员使用在改进的前景文档中定义的特性集来派生用例并以说明，这些用例详细阐述用户对系统预期行为的观点。用例模型作为客户、用户和系统开发人员之间的合同，规定了所选特性如何在系统中发挥作用。它有助于系统开发人员设置符合现实的期望和目标，帮助客户和用户验证系统是否达到了这些期望。

一些系统需求没有很好地符合用例。系统分析员就在补充规约里说明这些需求。许多非功能性需求，如可用性、可靠性、性能、可支持性等，都放在补充规约中。应该注意，这些类型的许多非功能性需求专门针对单个用例。用例阐释者最好将这些需求放在用例规约本身（请参见改进系统工作流），在补充规约里描述系统范围的非功能性需求。

在该工作流明细中，系统分析员创建和设置了补充需求的属性（如优先级和相关用例）。除此之外，系统分析员还为初始用例和新用例添加并更新属性值。

最后，系统分析员通过追踪重要用户需要和关键特性到相关用例以及补充规约说明的需求，以此来管理依赖关系。

工作流明细：管理系统规模

在确定特性级别的需求，描述大多数主角、用例以及补充规约所指定的其他需求后，,系统分析员应该收集需求属性（如优先级、工作量、成本和风险等），并尽可能精确地为其指定属性值。这使得人们对如何确定系统发布的初始规模有了更好的理解，也让构架设计师能够从结构上确定具有重要意义的用例。

由项目和开发管理人员一起制定的迭代计划，首次出现在该工作流明细：管理系统规模中。迭代计划也是一个开发计划，它定义为发布版计划的迭代次数和频率。早期的迭代应计划规模内风险最高的元素。

管理规模工作流的其他重要输出包括软件构架文档的初始迭代和一个修订过的前景文档，前景文档反映分析员和关键项目干系人对系统功能和项目资源的加深理解。

与前面提到的商业理由一样，迭代计划的第一个问题是，软件构架文档不是需求管理作流程的一个工件，尽管它与该工作流有关而且是 Rational Unified Process 的一部分。这部分内容不在本文的讨论范围内。

> 经验告诉我们，成功管理规模的关键在于，仔细考虑分配给项目干系人需要、特性、用例和补充规约所指定的其他需求的属性值，与代表性项目干系人定期进行开放而诚恳的交流。

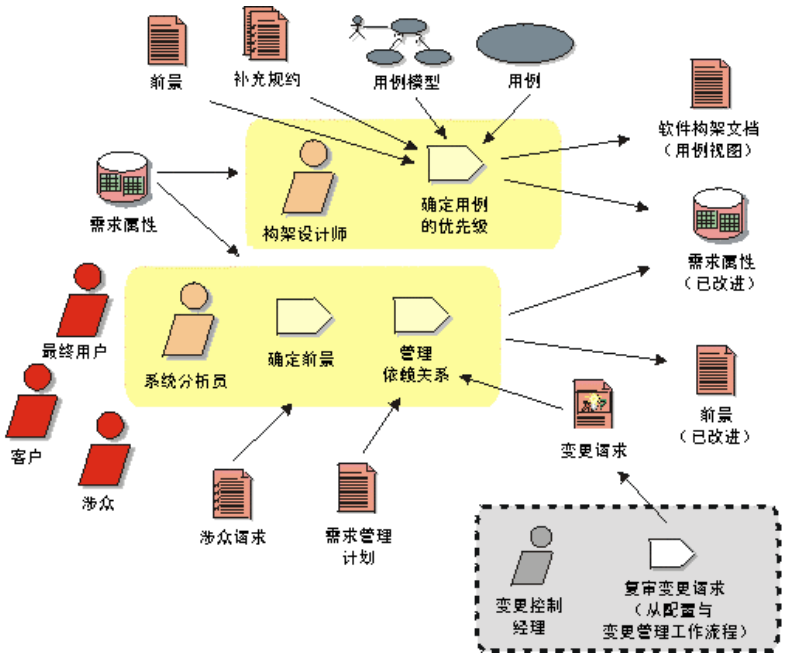

图 13 - 管理系统规模

工作流明细：管理系统规模

构架设计师为用例的风险范围、构架重要性和构架覆盖范围确定优先顺序。尽管定义系统时用到了许多用例和补充规约需求，但通常只有用例的一个子集才是好的系统构架的关键。确定用例的优先级后，构架设计师改进迭代或开发计划，利用 Rational Rose 这类工具建立系统构架的用例视图模型。

系统分析员通过改进前景中特性的需求属性来管理依赖关系。他们还对用例和补充规约里的需求进行改进。系统分析员确保项目干系人需要、特性、用例需求和补充规约需求存在适当的可追踪性。这里特意使用了“适当的可追踪性”。请参见本文中插入的关于可追踪性的说明文字。

在整个项目中，该步骤是最为重要的步骤之一。第一次，我们对所提议的系统了解如此之深，使得我们能对需求、项目资源和交付日期作出重大承诺。同时也必须理解，这些需求将会随着了解程度的加深而变更。如果在前三个工作流对依赖关系进行管理，则执行这一步骤就会容易得多，将来进行变更也更容易。

贯穿项目的整个生命周期，随着形势和环境的变化，由于系统分析员必须就修改后的项目规模和前景与关键项目干系人进行协商，因此将会多次重新检查该工作流明细。项目干系人和开发团队成员只有把该步骤看作一个自然前进过程 - 既不要让用户措手不及，也不要企图向组织索要更多时间和金钱 - 才能成功管理项目规模，使之与可用资源相符合。该工作流在项目的主要里程碑处需重复多次，以便评估对系统及系统问题的新认识是否要求变更规模。尽管已承诺的需求、预算和期限很难更改，但随着对确定优先级的用例、补充规约和早期系统迭代的深入理解，不可避免会导致人们重新考虑系统规模。

需要重申，在到达改进阶段之前，以及在贯穿项目生命周期内发生变化前，项目团队应维持日常的规模管理，这一点很关键。项目干系人代表必须理解并相信，在难度不断增加的规模协商中，他们的优先考虑和利益始终得到认真的对待。在改进系统需求之前，只有重要的需求才有待于协商或修改。除非建立有效的规模管理制度，否则项目注定会变成一次“死亡之旅” - 规模过大的项目将残酷地走向延期和成本超支的绝路。

工作流明细：改进系统定义

进入改进系统定义后，该工作流明细假设已经概述了系统级别的用例并至少简要描述了主角。通过项目规模管理，前景中的特性再次确定了优先顺序，现在可以相信，这些特性在比较严格的预算和期限下是可以实现的。该工作流的输出是对系统功能更深入的理解，这些系统功能在详细用例、已修订的补充规约以及系统本身的初期迭代中进行说明。

显然，不是所有的系统都有用户界面，不是所有的初期迭代都包括 GUI 元素。这里我们仅仅将它们作为初期迭代的示例。其他例子包括原型、模型和示意板。

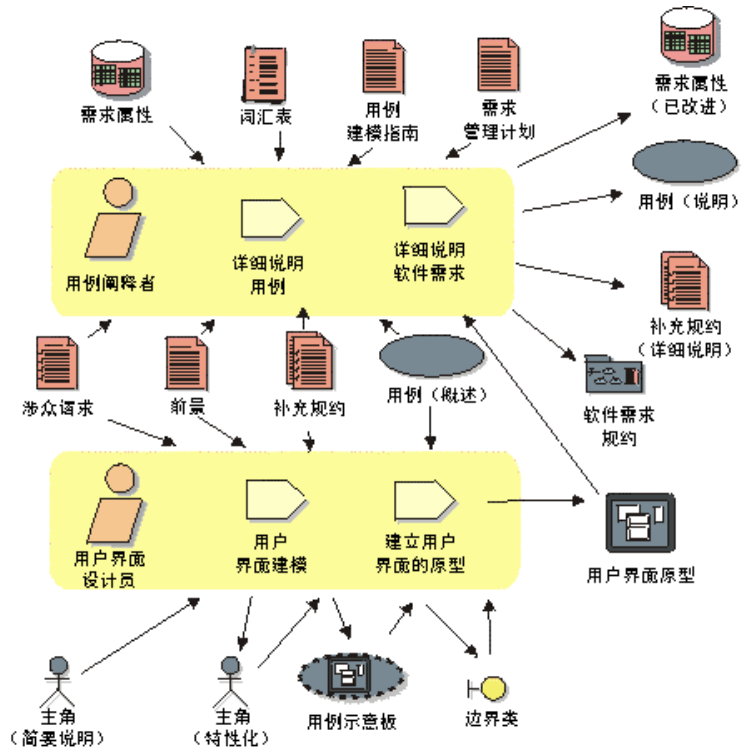

图 14 - 改进系统定义

工作流明细：改进系统定义

用例阐释者详述每个用例的事件流定义、前置和后置条件以及其他文本属性。为最大限度地减少工作量并提高可读性，建议使用标准文档格式或用例规约模板来记录每个用例的文字信息。创建考虑周全的用例规约对系统质量至关重要。制定规约要求对项目干系人需要以及 needs and features 与用例相关的特性有透彻的理解。让项目团队的若干成员（如软件工程师）参与用例创建是再好不过了。

同时，用例阐释者使用并非用例特有的附加需求对补充规约进行修订。

用户界面设计员模拟系统的用户界面并进行原型设计。这项工作与用例的演进密切相关。

对每个需求有更深入的理解后，用例阐释者和系统分析员对其工作量、成本、风险以及其他属性值进行修正。

系统定义改进流程的结果提交给另一轮管理规模工作流明细。对系统有更多的了解后，可能需要改变优先级。毫无疑问，如果必要，系统发布的规模将需要复审和改进。

工作流明细：管理需求变更

当变更发生时 -- 变更是不可避免会发生的 -- 管理需求变更工作流明细需持续应用于项目生命的全过程，这与管理系统规模工作流明细一样。该工作流的输出可能导致对每个工件的修改，这又要求在所有的项目团队成员和项目干系人之间进行有效的交流。

在这个工作流中，我们引入了受需求工作流影响的其他工件。需求的变更必然影响在分析设计工作流中表示的系统模型。需求变更还影响用于验证需求是否正确实施的测试。在前面的例子中，这些工件是 Rational Unified Process 的组成部分，但不是本文论述的主题。在管理依赖关系流程中确定的可追踪性关系是理解这些影响的关键。

可追踪性

在需求领域，与可追踪性有关的事务有很多。许多事务都具有追踪单个客户需求到每个相关规约、测试、模型元素以及最终源码文件的特点。的确，一些可追踪性是成功的需求变更管理的关键。

然而，这里事先警告，在项目的整个生命周期中，建立和维护各种形式的可追踪性都需要投资。像所有的投资一样，可追踪性的投资回报点逐渐减少，这取决于具体情况。本文强调在不同需求类型之间进行追踪的价值。这是一个很好的起点，而且可以用 Rational RequisitePro、Rose、SoDA 和 TeamTest 这样的工具使之自动化。我们相信，你终将发现某个级别的需求可追踪性是好的投资对象。

要了解需求可追踪性策略的更多信息，请参见白皮书“通过用例进行需求管理的可追踪性策略” [6]。

管理需求变更工作流的另一个重要概念是需求历史追踪。通过把握需求变更的性质和基本原理，复审员（工作受变更影响的任何软件项目团队成员）将收到对变更作出正确响应所需的信息。

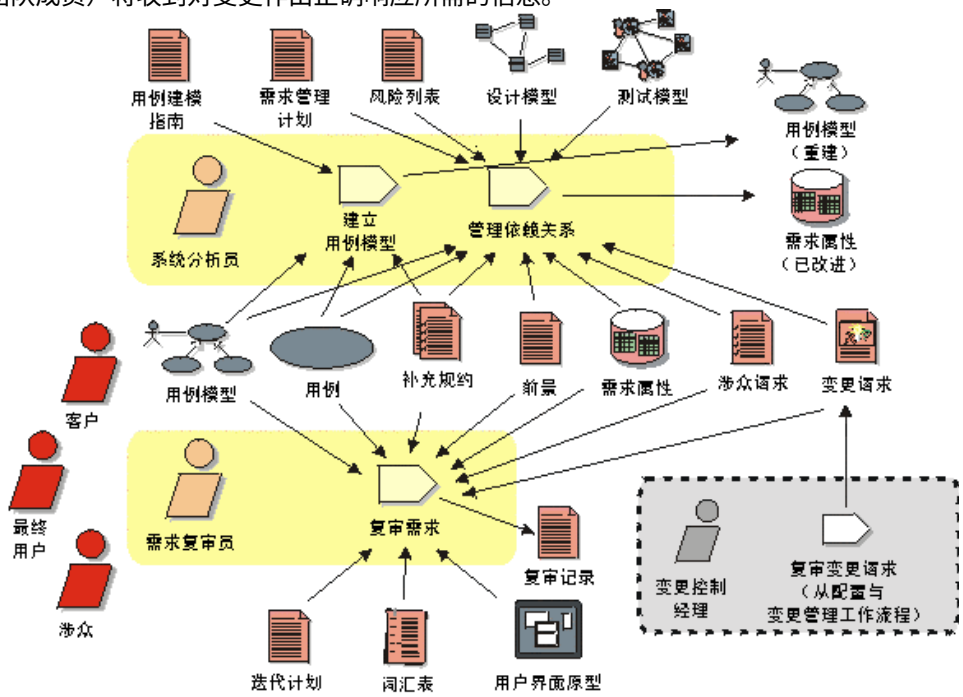

图 15 - 管理需求变更

工作流明细：管理需求变更

出于各种原因，任何项目干系人或项目团队成员都有可能提出变更需求的请求。所有变更请求(Cr)，包括对需求或扩展请求甚至缺陷的变更，都应该通过同一个变更请求管理(CRM)流程进行疏导。至少，这应包括在一个集中数据库系统中记录并追踪请求，并由中央复审委员会执行复审。CRM流程的详情见 Rational Unified Process 的其他小节。

系统分析员应该协调最好由变更控制委员会(CCB)执行的复审活动，收集并检查所有的变更请求，并将它们分类为：

- 实施中不影响需求的缺陷，
- 对现有的某类型需求的修改，或者
- 扩展修改。

分类后，按在其他需求工作流中描述的方法为所提出的需求变更指定属性和值。

在复审变更请求的时候，系统分析员向一个由项目干系人代表和项目团队成员组成的 CCB 陈述确定优先顺序的需求变更。超过资源限制的规模修改请求被拒绝，或者将其上交给项目干系人代表，项目干系人代表有权批准对日期以及预算事项的必需变更。

CCB 批准或拒绝需求变更。

系统分析员把需求变更传达给需求阐释者，或直接修改前景、用例、补充规约文档或其他需求工件中的需求。

需求复审员（开发人员、测试员、经理及其他团队成员）通过审查需求历史，对需求变更对他们的工作造成的影响进行评估。最后，他们实施变更，对他们有权修改的相关需求作适当改动。

总结

管理需求的需要并不新鲜。那么，究竟是什么值得我们去思考呢？

首先，如果项目常常不能满足客户、错过最后期限和预算超支，就有理由重新考虑开发方案了。在设计方案时，如果您确信与需求相关的问题正在消耗你的开发工作，就有理由考虑改进需求管理了。

其次，本文总结的需求管理方案体现了几千个案例的集体经验和智慧，而且代表了许多个人深思熟虑的观点，他们在需求管理领域与客户合作已有数年时间。我们认为，他们的贡献 - 以及 Rational Unified Process 对这些贡献所作的透彻陈述 - 总结起来代表了需求管理的“最佳方案”。你将发现这些需求建议是最可靠的。

作者向 Dr. Ivar Jacobson 对本文的直接和间接贡献，以及 Dean Leffingwell、 Dr. Alan Davis、Ed Yourdon 和 Elemer Magaziner等人的帮助表示感谢。尤其，我们要感谢 Rational Software 的客户，他们在数以百计开发项目中应用和改进了这些方案。

最后，在过去的两年内，生产有效软件开发解决方案的领先公司 Rational Software 接受需求管理的挑战，生产了多种工具来使执行这一艰巨任务实现自动化。长期、普遍存在的需求管理问题得到了解决。也许这最终才是今天在需求管理领域开始追求卓越的最佳原因。

关于上述概念的完整论述，请参考 Dean Leffingwell 的关于软件需求管理的优秀著作[7]。

参考资料

[1] CHAOS, The Standish Group International, Inc., Dennis, MA, 1994, 1997. [2]

Computer Industry Daily, December 12, 1997.

[3] Dorfman, M. and R. Thayer, *Software Engineering, IEEE* Computer Society Press, Los Alamitos, CA, 1997 pp. 79.

[4] Dorfman, M. and R. Thayer, *Software Engineering,* IEEE Computer Society Press, Los Alamitos, CA, 1997 pp. 80.

[5] Spence, Ian and Leslee Probasco, “通过用例进行需求管理的可追踪性策略”， 白皮书, Rational Software Corporation, 1998.

[6] Rational Unified Process™, Rational Software Corporation, Cupertino, CA, 1999.

[7] Leffingwell, Dean and Don Widrig, Managing *Software Requirements - A Unified Approach,* , Addison-Wesley, 2000.

Rational
the software development company

两家总部：

Rational Software
18880 Homestead Road
Cupertino, CA 95014
电话：(408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
电话：(781) 676-2400

免费电话：(800) 728-1212
电子邮件：info@rational.com
Web: www.rational.com
全球网址：www.rational.com/worldwide