IBM Rational

# 需求管理

# IBM Rational 技术白皮书

版本 1.0

Rational. software

IBM

# 目录

Rational. software

IBM

# 需求管理白皮书

## 需求和需求管理

### 为什么需要管理需求?

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 2001 年的 CHAOS Reports 证实，导致项目失败的最重要的原因与需求有关。
- 2001 年，Standish Group 的 CHAOS Reports 报导了该公司的一项研究，该公司对多个项目作调查后发现，百分之七十四的项目是失败的，既这些项目不能按时按预算完成。其中提到最多的导致项目失败的原因就是“变更用户需求”。

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。
- 系统或系统构件为了满足合同、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。”

### 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。这里介绍的以及 IBM Rational 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涉众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涉众请求，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。

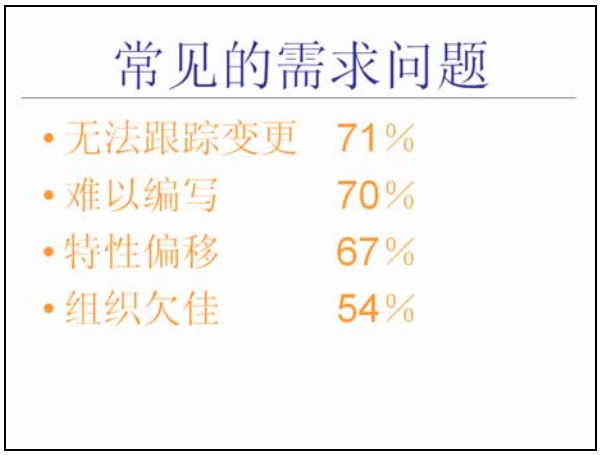



图 1 - 常见的需求问题

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工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。IBM Rational 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,并使用相应的工具使得上述的流程和专业技术得以实现。

## 需求捕获

从上述的分析可以看出，需求的捕获是需求管理的基础和前提。在这里，将介绍一种为业界所广泛采用并经验证的需求捕获方法，即用例模型。

用例模型是系统既定功能及系统环境的模型，并作为客户和开发人员之间的契约。用例模型用作分析、设计和测试活动的基本输入。用例是贯穿整个系统开发的一条主线。同一个用例模型即为需求工作流程的结果，可当作分析设计工作流程以及测试工作流程的输入使用。参与者(Actor)和用例(UseCase)是用例模型中的主要元素。

下图显示了自动取款机系统用例模型的一部分:

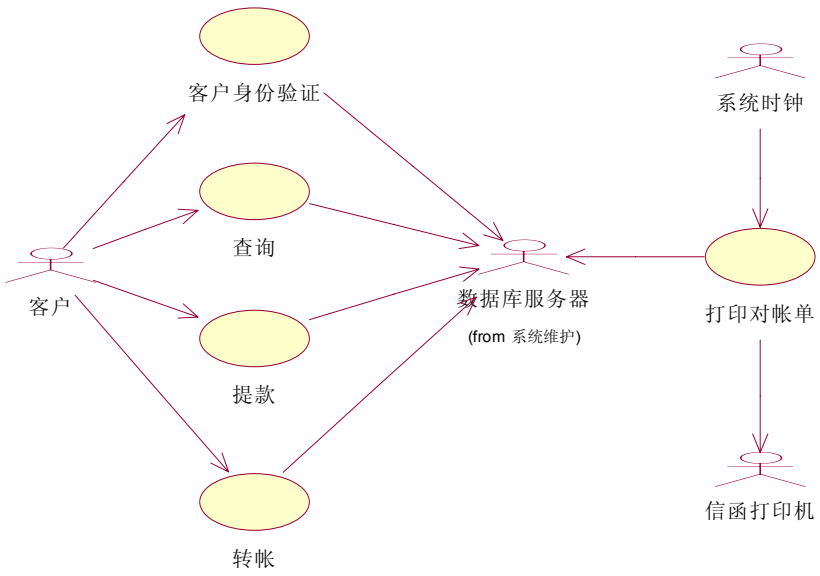



图 2 - 用例模型示例

用例图用于显示包含参与者和用例的用例模型示例。系统建模有许多种方法，每种建模方法可以满足不同的目的。然而，用例模型最重要的作用是将系统行为传达给客户或最终用户。可能与该系统交互的用户和任何其他系统都是参与者。由于参与者代表了系统用户，它们协助界定系统并提供十分明确的系统用途说明。编写用例依据参与者的需求来进行。这样就确保该系统成为用户期望得到的系统。

参与者和用例都是通过将客户需求及潜在用户当作重要的信息查找到的。找到这些用例和参与者后，应对它们作简要说明。在详细说明这些用例之前，客户应复审该用例模型以核实所有的用例和参与者都已经找到，并且它们可以提供客户所需要的东西。

在迭代开发环境中，您可以选择用例的子集以便在每个迭代中详细描述。参与者和用例找到后，需要详细说明每个用例的事件流。这些说明指出系统与参与者交互的方式以及在各个独立用例中系统执行的有关操作。

最后，对已完成的用例模型（包括用例说明）进行复审，开发人员和客户使用该模型对系统应执行的操作达成一致意见。

## 需求管理模型

在需求管理的流程中，需求的捕获手段固然重要，但在需求的捕获和需求最终成型的过程中，我们会面临各种和需求相关的信息和资料（也可以把这些信息笼统地称做“需求”），如何发现这些信息之间的关系并有效组织，更为关键。

### 需求类型

在 RUP 中，我们采用一种金字塔方式的管理办法，来组织和管理我们获取的信息乃至最终的需求。

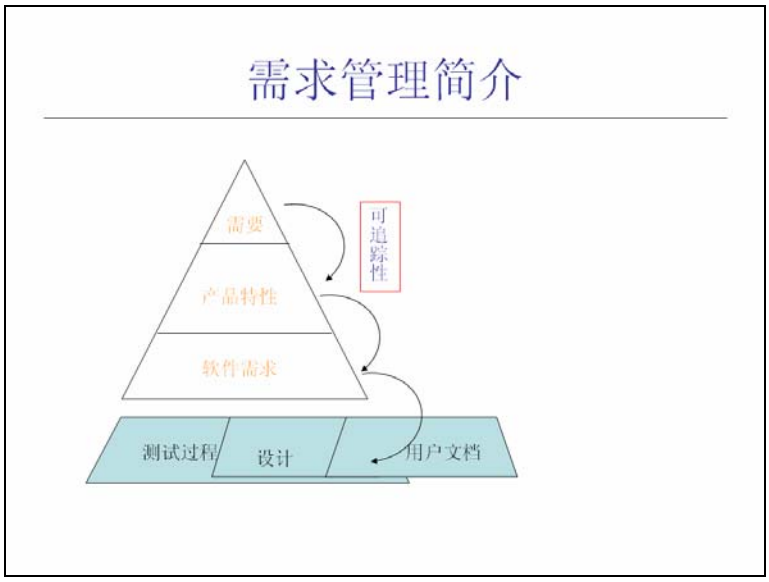


图 3 - 需求管理结构概述

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，团队必须确定涉众，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。而后，由产品特性来抽取软件需求，在我们的模型中，软件需求是以用例模型的方式来描述。从测试的角度来看，测试项一定来自于软件需求，即软件需求中确定了哪些需求项，测试就要根据这些需求项来制定和实现。

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。从上述的分析中我们可以看到，通常，一类需求可以细分即分解成其他类型的需求。这里，我们就把需求分解为几种类型，并在他们之间建立相应的关联。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。上述的这些需求类型同时保存在对应的 RUP 文档和数据库中。

**应用需求类型**

通过定义需求类型，以及他们之间的关系，我们就建立了一个需求管理模型的框架。当然，我们建立这样的一个模型，是为了方便我们使用需求，为了达到这一目的，我们还需要在此基础上添加相应的内容。

需要对各种需求类型添加它们的属性，以便于对需求进行查询等管理手段。比如，可以针对用户需要，确定该需要的必要性、优先级、确定性等属性。在实际的项目中，就可以确定这些属性的值，而后根据这些实际属性值来安排项目的进度表等。或是在项目进度紧急时，确定哪些需求是可以延期完成，而哪些是必须完成的，等等。

**需求的追踪性**

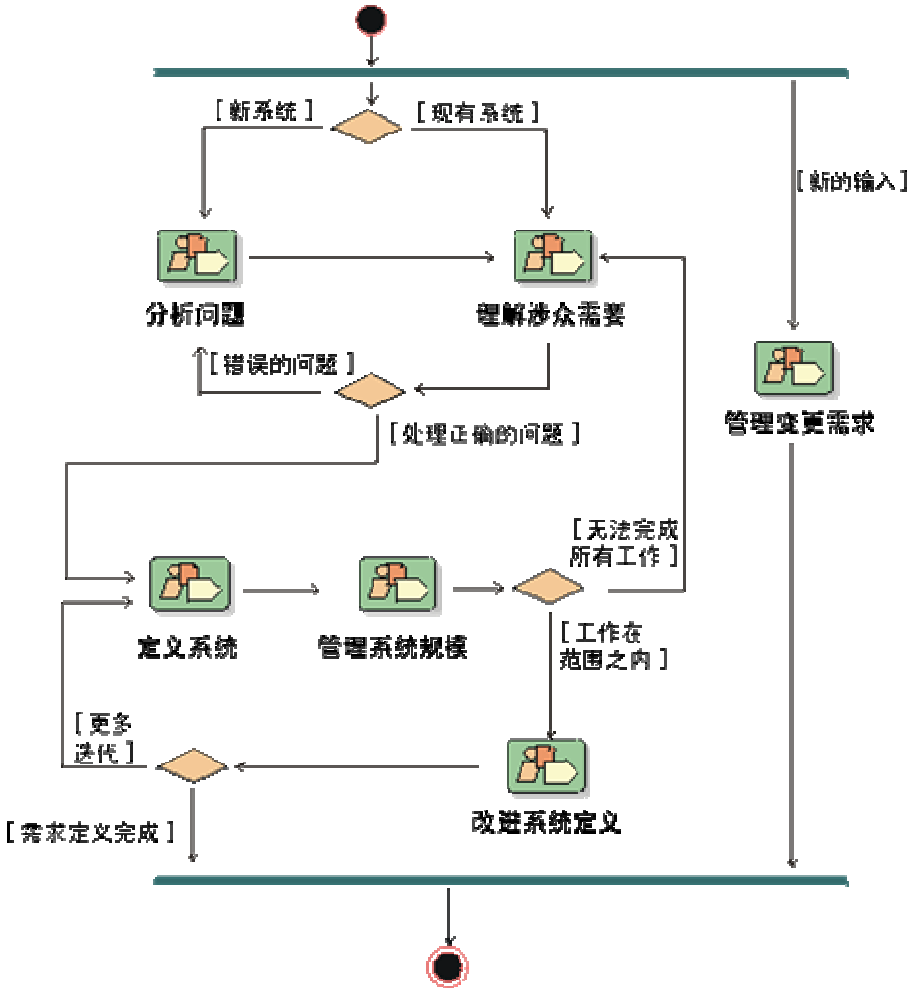
其次，可以根据不同需求的导出情况，在不同的需求之间建立追踪关系。譬如，用户需要决定了要构建产品的特性，产品的特性又决定了产品的软件需求，等。在这些不同类型的需求之间建立关联，一旦其中的某些需求发生变化，就可以确定它可能带来的影响，从而制定相应的策略。

Rational. software

IBM

# 需求管理的工作流程

## 需求管理的工作流程




## 工作流明细简介

*问题分析*

问题分析可以通过了解问题及涉众的最初需要，并提出高层解决方案来实现。它是为找出“隐藏在问题之后的问题”而进行的推理和分析。问题分析期间，将对“什么是面临实际问题”和“谁是涉众”等问题达成一致。而且，您还要从业务角度界定解决方案，以及制约该解决方案的因素。您应该已经对项目进行过商业理由分析，这将便于您更好地预计能从构建中的项目中得到多少投资回报。

*理解涉众需要*

需求来自各个方面，比如来自客户、合作伙伴、最终用户或是某领域的专家。您需要掌握如何准确判断需求应来源于哪方面、如何接近这些来源并从中获取信息。提供这些信息主要出处的个人在本项目中称为涉众。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。通常讨论将在业务模型这一级上展开，而不是在系统这一级上展开。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。

获取需要的活动可使用这样一些技巧：访谈、集体讨论、概念原型设计、问卷调查和竞争性分析等。获取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。

*定义系统*

定义系统指的是解释涉众需求，并整理为对要构建系统的意义明确的说明。在系统定义的初期要确定以下内容：需求构成、文档格式、语言形式、需求的具体程度（需求量及详细程度）、需求的优先级和预计工作量（不同人在不同的实践中通常对这两项内容的看法大不相同）、技术和管理风险以及最初规模。系统定义活动还可包括与最关键的涉众请求直接联系的初期原型和设计模型。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。

*管理项目规模*

为使项目高效运作，应仔细根据所有涉众的需求确定优先级，并对项目规模进行管理。有的开发人员仅仅重视所谓的“复活节彩蛋”（即开发人员感兴趣或觉得有挑战性的特性），而不是及早将精力投入降低项目风险或提高应用程序构架稳定性方面，这已使太多的项目蒙受损失。为确保尽早解决或降低项目中的风险，应以递增的方式开发系统。要慎重选择需求，以确保每次增加都能缓解项目中的已知风险。要达到目的，您需要和项目的涉众协商每次迭代的范围。通常，这要求具备管理项目各个阶段的期望结果的良好技能。除了控制开发过程本身，您还需控制需求的来源，并控制项目可交付工件的外观。

*改进系统定义*

系统的详细定义应能让涉众理解、同意并认可。它不仅需要具备所有功能，而且应符合法律或法规上的要求，符合可用性、可靠性、性能、可支持性和可维护性。感觉构建过程复杂的系统就应该有复杂的定义，这是一种常见的错误看法。这会给解释项目和系统的目的造成困难。人们可能印象深刻，但他们会因不甚理解而无法给出建议。应该致力于了解您制作的系统说明文档的读者。您可能常会发现需要为不同的读者准备不同的说明文档。

我们认为用例方法是传达系统目的和定义系统细节的一种行之有效的方法，它常与简单的可视化原型结合使用。用例有助于为需求提供一个环境，利用它可生动地说明系统使用的方式。

系统详细定义的另一个构件是说明系统采用的测试方式。测试计划及要执行测试的定义将会说明要核实哪些系统功能。

*管理需求变更*

定义需求时无论怎样谨慎小心，也总会有可变因素。变更的需求之所以变得难以管理，不仅是因为一个变更了的需求意味着要花费或多或少的时间来实现某一个新特性，而且也因为对某个需求的变更很可能影响到其他需求。应确保赋予需求一个有弹性的结构，使它能适应变更，并且确保使用可追踪性链接可以表达需求与开发生命周期的其他工件之间的依赖关系。管理变更包括建立基线、确定需要追踪的重要依赖关系、建立相关项之间的可追踪性，以及变更控制等活动。