计算机科学与工程学院实验报告

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | | **需求工程** | | | **班级** | **18软工5班** |
| **作业名称** | | **我对需求分析与建模的认识与应有内容建议** | | | **教导教师** | **董瑞生** |
| **姓名** | **翁格婉** | | **学号** | **1814080902537** | **日期** | **2020.10.4** |
|  |  | |  | |  |  |

目录

[第一章 需求工程导论 1](#_Toc29626)

[1.1软件生产中的需求问题 1](#_Toc6699)

[1.1.1需求问题是当前软件开发面临的主要问题 1](#_Toc17853)

[1.1.2需求问题具体原因分析 4](#_Toc31627)

[1.1.3软件开发中碰到的的需求问题的现象是什么？ 7](#_Toc12544)

[1.1.4在需求处理当中要注意哪些非技术性因素，为什么？ 8](#_Toc19123)

[1.2需求工程 8](#_Toc21337)

[1.2.1需求工程的定义 8](#_Toc27736)

[1.2.2解释软件工程与系统工程之间的联系，这种联系对需求工程的工作有何影响？ 9](#_Toc11254)

[1.3需求工程师 9](#_Toc8959)

[1.3.1需求工程师需要具备哪些知识或技能？ 9](#_Toc23337)

[第二章 需求基础 9](#_Toc24070)

[2.1需求的定义 9](#_Toc2511)

[2.2需求的分类 10](#_Toc1542)

[2.2.1用户需求分类 10](#_Toc31900)

[2.2.2实例（文字处理系统部分软件规格说明书） 10](#_Toc22875)

[2.3任务问题与用户需求 10](#_Toc15119)

[2.4需求分析方法 11](#_Toc16847)

[第三章 需求工程过程 13](#_Toc12008)

[3.1需求工程过程 13](#_Toc5310)

[3.1.1需求工程过程的定义 13](#_Toc2122)

[3.1.2需求工程的各个活动，各自的工作基础、工作目标和工作成果 14](#_Toc31017)

[3.2需求获取 14](#_Toc21491)

[3.2.1软件需求包括 14](#_Toc19346)

[3.2.2需求获取方法与策略 15](#_Toc23599)

[3.2.3建立顺畅的通信途径 15](#_Toc21916)

[3.2.4访谈与调查 15](#_Toc29300)

[3.2.5.观察用户操作流程 15](#_Toc8130)

[3.2.6组成联合小组 16](#_Toc28358)

[3.2.7 FAST基本原则 16](#_Toc5535)

[3.2.8 FAST会议 步骤 16](#_Toc23381)

[3.2.9用况（Use Case） 17](#_Toc14749)

[3.3需求管理 17](#_Toc13814)

[引用文献 17](#_Toc19211)

# 第一章 需求工程导论

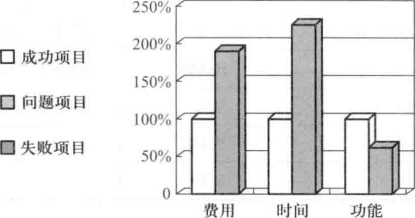
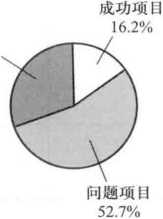
## 1.1软件生产中的需求问题

### 1.1.1需求问题是当前软件开发面临的主要问题

无论是实践者的切身体会，还是各种调查数据，都明确指出需求问题是当前软件开发面临的 主要问题之一。在所有调査数据中，以美国专门从事跟踪工厂项目成功或失败的权威机构 Standish Group的CHAOS系列报告最广为人知。

在Standish Gnwp的调査中将软件项目分为3种类别：

1. 在预计的时间之内，在预算的成本之下完成预期的所有功能，则项目为成功项目(suc­cess )o
2. 已经完成，软件产品能够正常工作，但在生产中或者超支，或者超期，或者实现的功能不 全，则项目为问题项目(challenged or faulty)。
3. 因无法进行而被中途撤销，或者最终产品无法提交使用，则项目为失败项目(failed or im­paired )o

Standish Group 1995年发布的调查报告［Standish 1995］表明(如图1-1和图1-2所示),1994 年美国365家公司的8 380个项目当中，成功项目仅为16.2%,失败项目为31.1%,问题项目为 52.7%。所有项目平均超支189%,平均超期222%,平均只完成了预计功能的61%。

失败项目

31.1%

口预期值

□实际值

图1-2项目质量情况调査, 数据来源于［Standish 1995］

为了更深入了解项目成败的原因.Standish Group在1995年的报告中还公布了导致项目成 功或失败的影响因素，相关数据如表1-1~表1-3所示。

表**1-1** 影响成功项目的因素，数据来源于[**Standish 1995]**

|  |  |  |  |
| --- | --- | --- | --- |
| 成功项目的影响因素 | 影响指数 | 成功项目的影响因素 | 影响指数 |
| 用户参与 | 15.9% | 员工能力 | 7.2% |
| 高层管理支持 | 13.9% | 主人翁精神 | 5.3% |
| 清晰的需求说明 | 13.0% | 清晰的目标和前景 | 2.9% |
| 正确的项目计划 | 9.6% | 努力工作 | 2.4% |
| 切合实际的期望 | 8.2% | 其他 | 13.9% |
| 细化的项目里程碑 | 7.7% |  |  |

表**1-2**影响问题项目的因素，数据来源于[**Standish 1995]**

|  |  |  |  |
| --- | --- | --- | --- |
| 问题项目的影响因素 | 影响指数 | 问题项目的影响因素 | 影响指数 |
| 缺少用户输入 | 12.8% | 不切实际的期望 | 5.9% |
| 不完整的需求说明 | 12.3% | 目标不清晰 | 5.3% |
| 需求变化 | 11.8% | 不现实的时间要求 | 4.3% |
| 缺乏高层管理支持 | 7.5% | 新技术的影响 | 3.7% |
| 技术能力不足 | 7.0% | 其他 | 23.0% |
| 缺乏资源 | 6.4% |  |  |

表**1-3** 影响失败项目的因素，数据来源于[**Standish 1995]**

|  |  |  |  |
| --- | --- | --- | --- |
| 失败项目的影响因素 | 影响指数 | 失败项目的影响因素 | 影响指数 |
| 不完整的需求说明 | 13.1% | 缺乏计划 | 8.1% |
| 缺少用户输入 | 12.4% | 额外的无用功能 | 7.5% |
| 缺乏资源 | 10.6% | 缺乏IT管理 | 6.2% |
| 不切实际的期望 | 9.9% | 技术能力不足 | 4.3% |
| 缺乏高层管理支持 | 9.3% | 其他 | 9.9% |
| 需求变化 | 8.7% |  |  |

通过分析表1-1~表1-3,可以发现需求因素对项目的成败具有至关重要的影响。其中的用 户参与（用户输入）、高层管理支持、清晰的需求说明、切合实际的期望、清晰的目标和前景、需求 变化、额外的无用功能等都会使需求发生问题。

1996年，欧洲软件协会（European Software Institute ,ESI）为欧洲软件过程改进培训计划项目 （European Software Process Improvement Training Initiative, ESPITI）发布的报告［ESPITI 1996 ］进 一步验证了 Standish Group的调查结果。

ESPITI在对欧洲17个国家的超过3 800个组织进行调查后发现，关于需求规格说明和需求 管理的缺陷是软件开发当中最常见的两类重要问题，如图1-3所示。

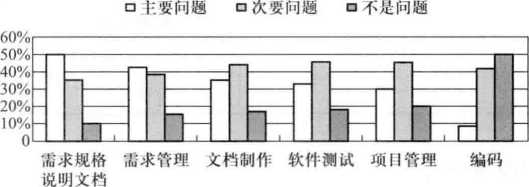


图1-3软件开发问题调査，数据来源于［ESPITI 1996］

所有这些调査数据表明，和软件需求相关的因素为软件项目所带来的风险和问题已经超过 了所有的其他因素，糟糕的软件生产状况背后隐藏着软件工程的需求问题。

到现在为止，［Standish 1995］和［ESPITI 1996］报告已经过去了近20年，其反映的问题有所 好转，但并未根本改变，软件生产面临的状况仍然不容乐观（如表1-4所示），其中的需求问题仍 然存在（如表1-5所示）。

表**1-4 Standish Group**报告中软件生产成功率的变化

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1994年度 | 1996年度 | 1998年度 | 2000年度 | 2004年度 | 2006年度 | 2008年度 | 2010年度 | 2012年度 |
| 成功/% | 16 | 27 | 26 | 28 | 29 | 35 | 32 | 37 | 39 |
| 问题/% | 53 | 33 | 46 | 49 | 53 | 46 | 44 | 41 | 43 |
| 失败/% | 3】 | 40 | 28 | 23 | 18 | 19 | 24 | 21 | 18 |

表**1-5 Standish Group**报告中项目成功的影响因素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 排序 | 2010年度 | | 2012年度 | |
| 影响因素 | 指数/% | 影响因素 | 指数/% |
| 1 | 用户参与 | 20 | 高层管理支持 | 19 |
| 2 | 高层管理支持 | 15 | 用户参与 | 18 |
| 3 | 清晰的业务目标 | 15 | 清晰的业务目标 | 15 |
| 4 | 情感成熟度（emotional maturity，即项目 氛围） | 12 | 情感成熟度 | 12 |

续表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 排序 | 2010年度 | | 2012年度 | |
| 影响因素 | 指数/% | 影响因素 | 指数/% |
| 5 | 最优化(optimization) | 11 | 最优化 | 11 |
| 6 | 敏捷过程 | 11 | 敏捷过程 | 9 |
| 7 | 项目管理技能 | 6 | 项目管理技能 | 7 |
| 8 | 有技能的员工 | 5 | 有技能的员工 | 5 |
| 9 | 执行力 | 3 | 执行力 | 4 |
| 10 | 工具与设备 | 2 | 工具与设备 | 1 |

### 1.1.2需求问题具体原因分析

软件生产中产生需求问题的最大原因在于对应用型软件的模拟特性理解不透彻或应用不坚 决，它会导致软件开发者产生轻视需求的态度问题。除此之外，还有一些技术原因也会导致需求 问题的产生。

1. 非技术性和社会性因素重视不足

应用型软件的模拟特性使得需求处理具有很突岀的特性。相对于软件开发的其他阶段而 言，需求处理阶段涉及更多的非技术性和社会性因素，并且其所受的影响也远远高于其他阶段。 20世纪90年代之前的需求处理往往更专注于技术处理，而对其中的非技术性和社会性因素重 视不足。

需求建模与分析是需求处理中的核心活动，它用一些形式化或半形式化的语言进行知识 的描述。一方面，只有通过建模与分析才能将混乱、模糊的用户需求变成清晰、明确的软件需 求，所以它是获取需求处理活动的必然后继，它建立的分析模型是需求处理中最为重要的成 果；另一方面，建模与分析的理论可以帮助人们系统化地看待问题，它可以根据理论或分析中 出现的各种现象指导其他需求处理活动更好地进行。因此，建模与分析活动在需求处理中具 有非常重要的地位，以至于人们理所当然地把需求处理工作的重心部署在建模与分析活动 中，放在对建模技术的理解和运用上，甚至在传统的软件开发生命周期中用“需求分析”一词 指代整个需求处理阶段。

但在需求处理阶段除了需求建模与分析活动之外，还有其他的活动也应得到重视，理解需 求处理中涉及的非技术性和社会性因素与理解建模分析技术一样必要，否则同样会导致软件 的失败，这些因素包括组织机构文化、社会背景、商业目标、利益协商等。它们的必要性具体 如下。

1. 从需求处理的任务来看，需要重视非技术性和社会性因素

需求处理的主要任务是发现问题并解决问题。现实是问题的发生地，软件系统是人们应对 问题的手段.但是单纯的软件系统是不能解决问题的，它只有和现实之间形成一种有效的互动 才能解决问题。因此，相对于软件系统的构造问题，人们更应该关注软件系统和现实之间的互动 效应。也就是说，需求处理不应该以新系统的功能性和内在特征为主要处理目标，而是更应该集 中精力于分析环境的构成、现状和它们将来能与软件达成的期望互动效应。因此，作为软件系统 环境的组织机构文化、社会背景和系统涉众(stakeholder,是指将会受到软件系统的影响，并能够 直接或间接影响系统需求的个人、团体或组织)的目标与利益比软件内部的数据流与状态更应该 得到重视。

1. 从需求处理的手段来看，需要重视非技术性和社会性因素

建模与分析技术是进行需求处理的主要手段，这些技术本身都是概念性的，不依赖于某些特 殊的应用环境条件，可以被广泛应用于各种应用场景。但是利用这些技术构建的解决方案一定 是和具体应用环境相关的，不存在不依赖于具体应用环境的解决方案。因此，在利用建模与分析 技术进行需求处理时，不能忽视具体应用环境中的相关因素，例如组织机构的文化、行业规范和 社会背景等，都会约束解决方案的构建空间。

1. 从需求处理的过程来看，需要重视非技术性和社会性因素

在需求处理的过程中，试图单纯通过技术的运用来建立一个一致、完整的需求模型是不太可 能的。因为在现实中，因涉众的不同立场而产生利益冲突的情景非常常见，这些冲突是根本性 的，是无法单纯通过技术手段所能解决的。因此，在需求处理的过程中，要重视非技术性和社会 性因素所导致的问题的解决，面对冲突要能够分析社会原因和组织机构方面的原因，引导涉众进 行利益协商，进而建立一个一致、完整的需求模型。

1. 传统需求分析方法的缺陷

传统的需求分析方法，如结构化分析和面向对象分析，都是从设计领域转入分析领域的。虽 然它们在设计阶段取得了很大的成功，但它们并不非常适合于需求阶段的技术处理需要，因此它 们在需求处理中的应用具有一定的先天缺陷。

传统的结构化方法和面向对象方法都是最先在编程领域取得成功的，它们所用的概念和组 织机制都是从编程领域抽象出来的。其后，它们又都相继被用来进行软件设计，因为设计和编程 都有构建高质量(健壮性、可维护性、适应性等)软件的共同目标，而且使用相同的概念和组织机 制保证了从设计到编程的平滑过渡，所以它们在设计领域的应用也取得了成功。而后它们又被 进一步应用到分析领域，但是需求分析除了拥有构建高质量软件的目标之外，还有一个更加重要 的目标是理解现实，而这是传统分析方法所拥有的概念和组织机制所无法实现的，所以说传统分 析方法在需求分析领域的应用具有先天缺陷。

1. 软件规模的日益扩大

20世纪90年代之后大量出现的以“企业”为中心的软件反映了软件规模日益扩大的发展趋 势，这一方面提高了需求处理中非技术性和社会性因素的影响比重，另一方面也进一步放大了传 统技术在需求处理阶段的不适应性。

在软件以单一任务或几个相关任务为应用领域时，软件应用的上下文环境相对局限在某个 部门或者某个角色，甚至某个个人的任务范围之内，涉众非常有限。所以，它所涉及的组织机构 文化、社会背景、商业目标和利益协商等非技术性因素自然也相对较少。而且该类软件的需求来 源往往很有限，所以每条需求相对较为完整和一致，可理解性相对较好，进行技术分析时对“为什 么做”(why)进行描述的要求不是非常必要。

在软件以企业为中心时，很少有用户能够单独给出对全局的理解，进而得出需求。相反，每 个用户往往仅能给出与其相关的片段，需要分析人员将所有用户的片段连接起来，构成全局理 解，导出需求。因此，该类软件要求分析人员能够在拥有相对有限的用户描述片段或者用户描述 片段间有冲突时进行相对正确的解读，即需求分析对规格说明可理解性的要求加强，这样对“为 什么做”进行描述就显得非常重要，因此传统分析方法的缺陷也就更加明显。

1. 需求问题的高代价性

需求处理是软件工程的起始阶段，设计、实现等后续阶段的正确性都以它的正确性为前提。 如果在需求处理过程中有错误未能解决，则其后的所有阶段都会受到影响，因此与需求有关的错 误修复代价较高，需求问题对软件成败的影响较大。统计表明，在需求阶段发生的错误如果到了 维护阶段才发现，则在维护阶段进行修复的代价可能高达需求阶段修复代价的100~200倍 [Boehm 1981, Boehm 2001 ]（如图1-5所示）。这种递增效应也说明了需求问题的高代价性。

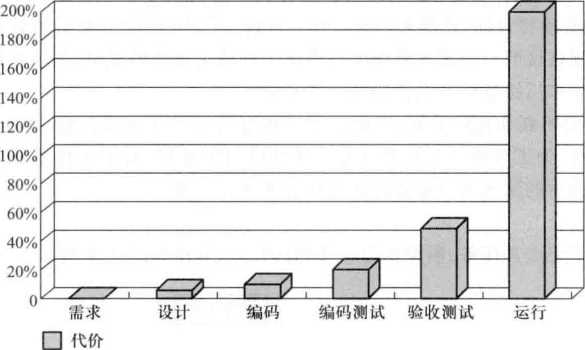


图1-5需求错误的修复代价对比，数据来源于[Boehm 1981]

### 1.1.3软件开发中碰到的的需求问题的现象是什么？

（1）用户参与度不够

（2）高层管理支持力度不够

（3）没有清晰的需求说明

（4）没有清晰的目标和前景

（5）期望不切合实际

（6）需求变化影响

（7）增加了无用的额外功能

### 1.1.4在需求处理当中要注意哪些非技术性因素，为什么？

1. 需求处理的任务：需求处理的任务主要是发现问题并解决问题。现实是问题的发生地，软件系统是人们应对问题的手段。但是单纯的软件系统是不能解决问题的。它只有和现实之间形成一种有效的互动才能解决问题。
2. 需求处理的手段：建模与分析技术是进行需求处理的主要手段，这些技术本身都是概念性的，不依赖于某些特殊的应用环境条件。可以被广泛的应用于各种应用场景。

（3）需求处理的过程：试图单纯的通过技术的应用建立一个一致完整的需求模型是不太可能的。因为在现实中，因涉众的不同立场而产生的利益冲突的场景非常常见。这些冲突是根本无法通过技术手段所能解决的。

## 1.2需求工程

### 1.2.1需求工程的定义

“需求工程”自产生以来，其概念和其领域内的其他名词一样，没有形成较为一致的定义，不 同的人从不同的角度出发，根据各自不同的理解，会得岀不同的定义。

简单来说，需求工程是所有需求处理活动的总和，它收集信息、分析问题、整合观点、记录需 求并验证其正确性，最终反映软件被应用后与其环境互动形成的期望效应。

从细节来看，可以定义如下：需求工程是软件工程的一个分支，它关注软件系统所应实现的 现实世界目标、软件系统的功能和软件系统应当遵守的约束，同时也关注以上因素和准确的软件 行为规格说明之间的联系，关注以上因素与其随时间或跨产品族而演化之后的相关因素之间的 联系。

通过以上定义可以发现，需求工程有以下3个主要任务。

①需求工程必须说明软件系统将被应用的环境及其目标，说明用来达成这些目标的软件功 能，还要说明在设计和实现这些功能时上下文环境对软件完成任务所用方式、方法所施加的限制 和约束，即要同时说明软件需要“做什么”和“为什么”需要做。

②需求工程必须将目标、功能和约束反映到软件系统中，映射为可行的软件行为，并对软件 行为进行准确的规格说明。需求规格说明是需求工程最为重要的成果，是项目规划、设计、测试、 用户手册编写等很多后续软件开发阶段的工作基础。

现实世界是不断变化的世界，因此需求工程还需要妥善处理目标、功能和约束随着时间 的演化情况。同时，为了节省开支和进行需求规格说明的重用，需求工程还需要对目标、功能和 约束在软件产品族中的演化和分布情况进行综合考虑与处理

### 1.2.2解释软件工程与系统工程之间的联系，这种联系对需求工程的工作有何影响？

（1）系统工程通常是指计算机引入某一现实系统，并用他来改变系统的运作方式，达到一个理想效果的过程。而且系统工程中除了含有处理系统的软件工程之外，还包括硬件工程和人力工程。因此，在系统工程中，虽然应该重点关注软件工程部分的内容，但并不能完全以软件为中心来看待和处理整个系统。

（2）影响：系统需求开发的主要目的是获得整个系统的期望目标，包括功能特性和非功能特性。因此需要判定系统的涉众，采集他们的目标与要求研究系统的环境确定系统的要求，并进行一些整体性的分析。

## 1.3需求工程师

### 1.3.1需求工程师需要具备哪些知识或技能？

1. 交流技能
2. 观察技能
3. 抽象分析与问题解决技能
4. 写作技能
5. 关系协调和团队工作技能

# 需求基础

## 2.1需求的定义

需求一直是软件工程中较为模糊的词汇之一。提起需求（requirement）,不同背景的人 （用户、开发者）会有不同的看法，因此需求是需求工程中一个非常难以准确定义和解释的 概念。

在各种不同的定义中，本书更倾向于使用IEEE的需求定义［IEEE 1990］：

1. 用户为了解决问题或达到某些目标所需要的条件或能力；
2. 系统或系统部件为了满足合同、标准、规范或其他正式文档所规定的要求而需要具备的 条件或能力；
3. 对①或②中的一个条件或一种能力的一种文档化表述。

IEEE的定义中同时包括了用户的观点（第一种条件和能力）和开发者的观点（第二种条件 和能力），它强调了 “需求”的两个不可分割的方面：需求是以用户为中心的，是与问题相联系的； 需求要被清晰、明确地写在文档上。

## 2.2需求的分类

### 2.2.1用户需求分类

1.目标需求：反映组织机构或客户对系统产品提出的高层次的目标需求，其限定了项目的范围和项目应达到的目标。

2.业务需求：主要描述了软件系统必须完成的任务、实际业务或工作流程等。

3.功能需求：指开发人员必须实现的软件功能或软件系统应具有的外部行为。

4.性能需求：指实现的软件系统功能应达到的技术指标，如计算精度、可靠性、可维护性和可扩展性瞪

5.约束与限制：指软件开发人员在设计和实现软件系统时的限制，如开发语言、使用的数据库等。

### 2.2.2实例（文字处理系统部分软件规格说明书）

1.目标需求：用户使用系统能有效地纠正文档中的拼写错误，系统能满足用户的业务要求以及提高用户的工作效率

2.业务需求：当找到文档中的拼写错误时，通过可供选择的单词表，选择单词表中的一个单词后，再替换掉原来的单词。

功能需求：查找文档中的单词，并高亮度地显示出错的单词。用对话框显示可供选择的单词表，实现整个文档范围内的替换。

3.性能需求：检查单词的速度快，准确率要求达到99%，系统的有效性和可靠性要高

4.约束与限制：文件内部格式要与word系统一致，开发平台为Linux系统，使用C语言

## 2.3任务问题与用户需求

高层次的目标是由组织的决策者提出的，但普通用户才是组织中任务的实际执行者，只有通 过一套具体并且合理的业务流程才能真正实现目标。用户需求就是执行实际工作的用户对系统 所能完成的具体任务的期望，描述了系统能够帮助用户做些什么。用户需求主要来自系统的使 用者——用户。在有些情况下，系统的直接用户不可知，如通用的软件系统或社会服务领域的软 件系统等，所以用户需求也可能来自间接的渠道，如销售人员和售后支持人员等。

例如，在超市管理系统中，收银员用户的需求如UR1~UR5所示。

UR1：收银员可以使用系统逐一记录销售的商品。

UR2：收银员可以使用系统计算商品账单并处理付款情况，账单计算需要使用促销策略。

UR3：收银员可以使用系统为顾客打印收据。

UR4：收银员可以使用系统退回顾客已经购买的商品。

用户需求是对任务的期望，所以其基本表达方式为“xx用户可以使用系统完成xx任务”。用 户任务应该是有目标性、有价值的活动。例如在银行ATM系统中，取款、存款、转账都是合理的 用户需求，因为它们各自代表了用户的一个目标，但“向ATM中插入银行卡”就不是一个合理的 用户需求，因为用户不会无目的地“向ATM中插入银行卡”。再如，UR1是一个合理的用户需 求，因为它表达了收银员的一个任务，但是“收银员使用扫描仪扫描商品条码”就不是合理的用 户需求，因为收银员的目的是记录商品，而扫描条码只是手段。

用户的任务可以有粒度不同的抽象表述，大的任务可以包含（分解为）小的任务，例如“销 售”是任务，可以分解为UR1~UR3。所以UR1-UR3是合理的，但UR4也是合理的。在实践中， 用户需求到底使用哪个粒度与抽象层次要依据软件系统的复杂度而定。本书建议使用UR1- UR3的粒度，它们的特点是无法再进行任务分解。但是在比较复杂的系统中，抽象度稍高的用户 需求也是可以接受的。还可以为用户需求建立嵌套层次结构，例如将UR1-UR3命名为UR5.1- UR5.3以表明它们是对UR5的细化。

UR5：收银员可以使用系统完成销售处理过程。

需要注意的是：在3个层次中，只有用户需求在表述时在不可验证性上要求较为宽松。因为 用户需求具有下面几个特点：

①模糊、不清晰。用户需求允许适度使用形容词和副词，使得描述常常带有模糊和不清晰 的特性。

②多特性混杂。在用户进行需求描述时，常常将功能需求和非功能需求混杂在一起。

③多逻辑混杂。用户需求是对用户任务的描述，而任务本身往往含有前后相继的多个逻辑 处理过程，即一个任务需要进行多次系统交互才能够完成。

用户需求表达了用户对系统的期望，但是要透彻和全面地了解用户的真正意图，仅仅拥有期 望是不够的，还需要知道期望所来源的背景知识。因此，对所有的用户需求都应该有充分的问题 域知识作为背景支持。而在实际工作中，用户表达自己的期望时，通常不会提及需求所涉及的问 题域知识，所以需求工程师需要根据用户的需求整理完整的问题域知识。例如对UR1,需要补充 问题域知识如下：

Data：需要记录的商品信息，包括ID、名称、描述、价格、特价、数量和总价。

Format：ID为xxx格式的条码。

Rule：总价=特价x数量。

## 2.4需求分析方法

需求分析有一套比较传统但好用的套路。我们可以从三个方面去分析需求：

1.角色

2.场景

3.路径

角色

需求最终使用者。

我们在分析角色时，不是分析人格，而是分析使用者在使用这个需求时，所扮演的角色。比如企业的管理者或者企业的员工。同一个真实个体可以隶属于多个角色。

在分析角色时，真正需要发掘和找寻的，是“人物”的动机。动机是角色背后最重要的因素。

举个例子，我走进饭店要了一碗饭，背后的动机就可能是“我饿了”。如何从角色（的行为）分析出动机，将严重影响需求的质量。

场景

需求被满足的场景。

通俗一点，我们在什么样的场景下需要满足这些需求。在通俗一点，角色在哪里用这个需求。

场景是需求很有用的一环。同样的角色，在不同场景下的需求是有变化。

举个例子，作为一个自然人，我在公司的穿着要求极有可能是正装，西装笔挺的迎接每天工作。但在家，我有可能穿着一件宽松的睡衣就可以到处跑。虽然都是“穿衣”的需求，但不同场景如何满足这些需求是不一样的。

路径

路径这个词比较绕口。我个人觉得如果是翻译成“步骤”会更好。

路径是需求被满足的步骤，也就是需求如何被满足的过程。

在分析需求时，有一个重点，就是“全路径”满足。所谓“全路径”满足，指的是需求在被满足的每个步骤都要满足用户的要求，而不仅仅是满足“结果”。

路径可能是一个非常漫长的，也可能是非常简短。要根据需求的粒度合理选择路径。

举个例子，用户购物这个需求的路径是：

选择产品->加入购物车->结帐->使用优惠券->支付->发货->用户收到货物->评价

这里就是一个非常长的路径。

需求决策

有了需求并对需求进行分析后，那么下一步就是对需求进行决策。

对需求进行决策是非常重要，也是非常困难的一步。该步骤中的核心只有一点“选出核心功能”。

在资源、精力、时间有限的情况下，为了让产品更有竞争力，找出核心功能并用所有资源尽力实现是产品成功的第一步。

找出核心功能没有相对通用的方法论。我们可以从以下几个角度分析：

1.竞品分析

2.公司战略

3.人力资源

4.时间

5.技术难度

6.产品自身定位

7.etc

除了竞品分析和产品自身定位外，其他思考点均为限制因素。针对限制因素，要关注的永远都是“平衡”，平衡所有可以平衡的资源与因素。而针对竞品和定位，则需要根据实际情况进行分析。

信息组织形式

为了给用户提供更方便的信息获取方式，打散与重组是必需的工作。

所谓打散，是将整块的信息打散成碎片化的信息，比如一篇博客，将其打散成博客标题、博客内容、博客作者、博客Tag等不同信息，从而方便进一步的信息利用。

而重组是将上述打散的信息内容进行重新组合，从而形成新的表现形式的内容。例如我们将博客标题进行列表展示，就为用户提供了博客列表这种新的信息展现形式，为用户提供了另一种信息获取的方式

# 需求工程过程

## 3.1需求工程过程

### 3.1.1需求工程过程的定义

过程是一组相关活动的集成，通过这些活动的执行，可以完成一项任务或者达到一个目标。 需求工程过程是系统开发中需求开发活动的集成，它以用户所面临的业务问题为出发点进行分 析和各种转换，最终产生一个能够在用户环境下解决用户业务问题的系统方案，并将其文档化为 明确的规格说明。

为了有效地理解用户问题，分析各种可能的系统解决方案，最终产生一个适宜的规格说明文 档，需要将需求开发活动组织成一个系统化和严格的需求工程过程，这是人们随着系统开发的进 展而逐渐认识到的［Siddiqi 1996］。在初期，系统开发的唯一焦点就是编码，此时不论系统大小， 开发都是一个单独的活动——编码。这个时期人们还不认为存在独立的需求开发活动。其后， 随着生命周期模型的引入，对系统开发活动的认知取得重大进展，人们认识到需求开发是系统开 发中的一个独立的阶段，即软件开发生命周期模型的第一个阶段。在此后的进一步发展中，人们 逐渐认识和接受了系统的演化式开发思想，认识到系统的实现往往是开始于一个并非完备的需 求体系，发现需求开发也是一个递进的过程，包含一系列的独立活动。今天，大多数软件专业人 士已经意识到需求工程也有属于它自己的生命周期模型，即存在针对需求开发的需求工程过程， 这个过程又作为系统工程和软件工程的一个子过程部署在系统开发的初期阶段。

目前看来，如果所开发系统的类型不同、开发公司的规模和文化不同，或者系统开发资源获 取的途径不同，需求工程过程都会表现出极大的差异。例如，对一个大规模的军用或航空系统而 言，通常存在一个正式、严格的需求工程过程，它详细定义了过程的各个阶段，要求产生能够详细 描述系统需求和软件需求的规格说明文档。而对开发创新型软件的小公司而言，需求工程过程 可能仅仅包含一些头脑风暴会议，它产生的文档也可能仅仅是对系统期望的一段简短描述。但 不管实际上执行的需求工程过程为何，它们都拥有一些共同的需求工程活动：需求获取、需求分 析、需求规格说明、需求验证和需求管理。其中，需求获取、需求分析、需求规格说明、需求验证为 需求开发活动，需求管理为项目管理活动。

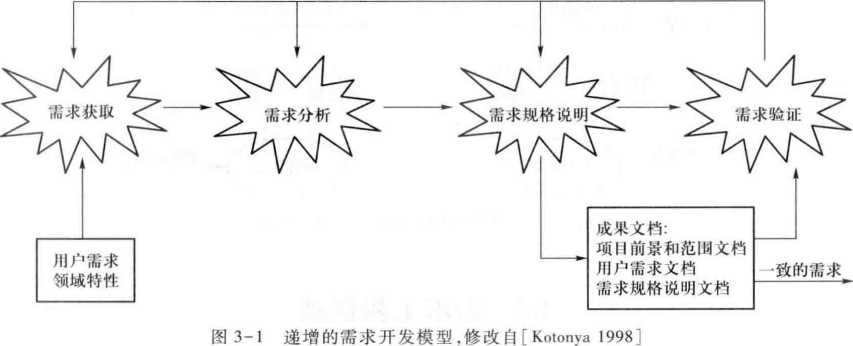
在需求工程的开始必须要获取用户期望系统表现出来的各种行为，这些期望并不是外在和 可以直接得到的，系统分析师需要利用一些方法和技术才能得到正确的结果。

为了取得对需求的正确和深入理解，系统分析师还需要对获取的结果进行综合与整理，通过 分析保证需求的正确性、完整性和可行性。

经过分析的需求被认为是有效的需求，它必须被记录和文档化，因为只有将需求文档化为正 式的规格说明，才可能将它们传递给其他参与系统开发的人员，让所有相关人员形成对系统需求 的一致和正确的理解。

规格说明文档可能被传递给设计人员、测试人员、项目管理人员等众多系统开发者，因此如 果传递的规格说明文档中存在一些错误或偏差将造成很大的影响。为了尽可能减小不利因素， 尽最大可能产生完善的规格说明文档，就需要在规格说明文档产生之后和传递给相关人员之前 组织进行文档的验证，以尽可能发现文档中的错误和偏差，并进行纠正。

获取、分析、规格说明和验证这些需求开发活动并不是看上去的那样以线性、顺序的方式执行。实 际上,这些活动之间是互相交织的，整个开发活动也是不断迭代和递增的（如图3-1所示）。



### 3.1.2需求工程的各个活动，各自的工作基础、工作目标和工作成果

（1）需求获取：（收集背景资料，获取问题与目标），（定义项目前景与范围），（记录获取结果）

（2）需求分析：（定义需求集合），（细化需求），（软件需求建模）

（3）需求规格说明：（获取的需求被编写文档），（系统涉众之间的交流信息），（编写文档）

（4）需求验证：（编写需求规格说明），（执行验证），（问题修正）

（5）需求管理：（建立维护需求基线集）（保证需求作用持续稳定和有效发挥）（随时进行变更控制）

## 3.2需求获取

### 3.2.1软件需求包括

功能需求

性能需求

用户或人的因素

环境需求

界面需求

文档需求

数据需求

资源使用需求

安全保密要求

可靠性需求

软件成本消耗与开发进度需求

其他非功能性要求

### 3.2.2需求获取方法与策略

建立顺畅的通信途径

访谈与调查

观察用户操作流程

组成联合小组

用况（Use Case）

### 3.2.3建立顺畅的通信途径

建立分析所需要的通信途径，以保证能顺利地对问题进行分析。

### 3.2.4访谈与调查

在具体的实践中，通常采用折衷的方法，即适当地计划好面谈，但不要过于详细，允许有一定的灵活性。一般按照如下原则进行准备：所提问的问题应该循序渐进，从整体的方面开始提问，接下来的问题应有助于对前面的问题更好的理解和细化；不要限制用户对问题的回答，这有可能会引出原先没有注意的问题；提问和回答在汇总后应能够反映用户需求的全貌。例子：“赛艇比赛成绩计算系统”的第一次面谈的准备计划。

### 3.2.5.观察用户操作流程

到用户的实际工作环境中对用户的工作流程进行观察，了解用户实际的操作环境、操作过程和操作要求，对照用户提交的问题陈述，对用户需求可以有更全面、更细致的认识。

### 3.2.6组成联合小组

便利的应用规约技术(FacilitatedApplication Specification Techniques , FAST) ：打破用户（需方）和开发者（供方）的界限，共同组成一个联合小组，发挥各自的长处，共同负责项目的推进，这样有助于发挥各自优势并增进解和协调

### 3.2.7 FAST基本原则

(1)在中立的地点举行由开发者和用户出席的会议；

(2)建立准备和参与会议的规则；

(3)建议一个足够正式的议程以便可以进行自由的交流；

(4)一个“协调者”(他可以是用户、开发者或其他外人)来控制会议；

(5)使用一种“定义机制”(它可以是工作表、图表、墙上胶黏纸或墙板)；

(6)目标是标识问题、提出解决方案的要素、商议不同的方法、以及在有利于完成目标的氛围中刻画出初步的需求。

### 3.2.8 FAST会议 步骤

1)当举行了开发者和用户之间的初步访谈后，确定一个FAST会议的时间地点，并在会议日之前将产品请求发布给所有的与会者。

2)要求每个FAST 出席者会前列出一组围绕系统环境的对象，以及对这些对象的操作或对象之间的交互功能，并开发出约束列表(如，成本、规模大小、权重)和性能标准列表(如，速度、精度)。这些列表可以不是穷尽的，但是，希望每套列表反映的是每个人对系统的感觉。

3)进行FAST 会议时，当团队的每个成员提出单个列表后，整个团队将创建一个组合的列表，该组合列表删去冗余项，并加入在表达过程中出现的新思想。在建好所有主题的组合列表后，开始讨论活动。缩短、加长或重新组合列表以适当地反映将被开发的产品。

4)一旦创建了意见一致的列表，应该将团队分为更小的小组，每个小组力图为每个列表中的一个或多个项开发出小型的规约（即对包含在列表中的单词或短语的精细化）。每个小组然后将他们开发的每个小规约提交给所有的FAST 出席者讨论，进行添加、删除或进一步的精化等工作。（在所有讨论过程中，团队可能提出某些不能在会议过程中解决的问题，此时要保留问题列表以使这些思想在以后的活动中产生作用。）

5)在小规约完成后，每个FAST 的出席者提出一个针对产品的确切标准列表，并将该列表提交给团队，然后创建一个意见一致的确定的标准列表。这个列表作为需求获取的结果，为需求分析和建模提供基础信息。

### 3.2.9用况（Use Case）

当需求作为非正式会议、Fast的一部分而收集起来之后，分析员就可以创建一组标识一串待建造系统的使用场景。

创建用况模型的主要步骤如下：

(1)确定谁会直接使用该系统，即参与者（Actor）

(2)选取其中一个参与者

(3)定义该参与者希望系统做什么，参与者希望系统作的每件事将成为一个用况

(4)对每件事来说，何时参与者会使用系统，通常会发生什么，这就是用况的基本过程

(5)描述该用况的基本过程

## 3.3需求管理

需求管理是一组用于帮助项目组在项目进展中的任何时候去标识、控制和跟踪需求的活动

需求跟踪有两种方式，正向跟踪与逆向跟踪

**正向跟踪：**以用户需求为切入点，检查《需求规约》中的每个需求是否都能在后继工作产品中找到对应点

**逆向跟踪：**检查设计文档、代码、测试用况等工作产品是否都能在《需求规约》中找到出处

# 引用文献

[Ahmed 2013 ] AHMED F, CAPRETZ L F, BOUKTIF Set. Soft skills and software development: a reflection from soft­ware industry. International Journal of Information Processing and Management(IJIPM), 2013, 4:3.

[Al- Ani 2006] AL- ANI B, SIM S E. So, you think you are a requirements engineer. 14th IEEE International Require­ments Engineering Conference (RE'06), 2006.

[Boehm 1981 ] BOEHM B W. Software engineering economics. Englewood Cliffs: Prentice Hall, 1981.

[Boehm 2001 ] BOEHM B, BASILI V R. Software defect reduction top 10 list. Computer Archive, 2001, 34( 1).

I. Brooks 1987] BROOKS F. No silver bullet: essence and accidents of software engineering. Computer, 1987, 4:10-19. [Capers 1996 ] JONES C. Applied software measurement: assuring productivity and quality. New York: McGraw -

Hill, 1996.

[Davis 2004] DAVTS A M. Great software debates. Wiley- IEEE Computer Society Press, 2004.

[ESPITI 1996] ESPITI project: european user survey analysis. European Software Institute (ESI), 1996.

[Francisco 2003 ] Francisco A C Pinheiro, Julio Cesar Sampaio do Prado Leite, Jaelson F B Castro. Requirements engi­neering technology transfer: an experience report. The Journal of Technology Transfer, 2003, 28(2): 159-65.

[Hofmann 2001 ] HOFMANN H F, Lehner F. Requirements engineering as a success factor in software projects. IEEE Software, 2001, 18(4):58-66.

[Jackson 1997 ] JACKSON M. The meaning of requirements. Annals of Software Engineering Special Issue on Software Requirements Engineering, 1997: 5-22.

[Juristo 2002] JURISTO N, MORENO A M, SILVA A. Is the european industry moving toward solving requirements engineering problems. IEEE Software 2002, 19(60)： 70-77.

[Katzenbach 1993] KATZENBACH J R,SMITH D K. The wisdom of teams. New York: Harper Business, 1993.

[Lamsweerde 2000] Lamsweerde V. A Requirements engineering in the year 00: a Research perspective. Invited Paper for ICSE, 2000-22nd International Conference on Software Engineering. Limerick: ACM Press, 2000.

[Lubars 1993 ] LUBARS M, Potts C, Richter C. A review of the state of the practice in requirements modeling. First Int\* 1 Symp.Requirements Eng. Los Alamitos: IEEE CS Press, 1993 : 2-14.

[Maiden 2007 ] MAIDEN N, NCUBEL C, ROBERTSON S. Can requirements be Creative? Experiences with an En­hanced Air Space Management System. 29th International Conference on Software Engineering (ICSE , 07), 2007.

[Minor 2004 ] MINOR O, ARMAREGO J. Requirements engineering: a close look at industry needs and model curricu­la. AWRE'04.

[Nikula 2000] NIKULA U, SJANIEMI J, Kalviainen H. A state- of- the- practice survey on requirements engineering in small- and medium- sized enteq)rises.Telecom Business Research Center Lappeenranta. Lappeenranta 2000.

[Nuseibeh 2000] NUSEIBEH B, EASTERBROOK S. Requirements engineering: a roadmap. Proceedings.of the 22nd Int' 1 Conference.on Software Engineering, Future of Software Engineering Track. New York: ACM Press, 2000.

[Oboler 2003] OBOLER A, SQUIRE D, KORB, K. Why don't we practice what we teach.Engineering Software for Computer Science Research in Academia. International Conference on Software Engineering Research and Appli­cations (SERA 2003), 2003.

[Paech 2008 ] PAECH B. What is a requirements engineer. IEEE Software, 2008, 7/8: 16-17.

[Penzenstadler 2009] PENZENSTADLER B, SCHLOSSER T, HALLER G, et al. Soft skills required: a practical ap­proach for empowering soft skills in the engineering world. Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills (CIRCUS 2009), 2009.

[Peter 1969] NAURP, RANDELLB. Software engineering: report on a conference sponsored by the NATO science committee. Garmisch, Germany, 7-11 Oct, 1968. Scientific Affairs Division NATO, 1969.

[Pinheiro 2003 ] PINHEIRO FAC. Requirements honesty, requirements engineering. 2003, 8(3): 183-192.

[Robertson 2002 ] ROBERTSON J. Eureka! why analysts should invent requirements. IEEE Software 2002, 7/8: 22-24. [Standish 1995] The Standish Group. CHAOS, 1995.

[Standish 2003 ] The Standish Group. What are your requirements. West Yarmouth, MA: The Standish Group Interna­tional, 2003.

[Thomasa 2002 ] THOMASA T, SCHRODERB C. Developing the interpersonal and communication skills necessary fbr effective requirements engineering, AWRE , 2002.

I Wiegers 2003 ] WIEGERS K. So you want to be a requirements analyst. Software Development, 2003.

[Wieringa 2003] WIERINGA R. Methodologies of requirements engineering research and practice: position statement. 1st International Workshop on Comparative Evaluation in Requirements Engineering. Faculty of Information Tech­nology. University of Technology, Sydney, 2003.

[Young 2002] YOUNG R R. Effective requirements practices. Boston: Addison- Wesley, 2002.

[Zave 1997 ] ZAVE P. Classification of research efforts in requirements engineering. ACM Computing Surveys, 1997, 29 (4): 315-321.