计算机科学与工程学院实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | | **需求分析与建模** | | | **班级** | **软工五班** | |
| **实验名称** | | **我对需求分析与建模的认识以及应有建议** | | | **教导教师** | | **许聚洛** |
| **姓名** | **许聚洛** | | **学号** | **1814080902526** | **日期** | **2020.10.10** | |
|  |  | |  | |  |  | | |

目录

[1. 软件开发的主要问题 1](#_Toc53253737)

[1.1软件的模拟特性 2](#_Toc53253738)

[2. 需求问题具体原因分析 3](#_Toc53253739)

[2.1非技术性和社会性因素重视不足  3](#_Toc53253740)

[3. 需求工程简介 5](#_Toc53253741)

[3.1 定义 5](#_Toc53253742)

[3.2 需求工程与系统工程  6](#_Toc53253743)

[4．对应有内容的建议 10](#_Toc53253744)

1. 软件开发的主要问题

当前软件开发面临的最大问题之一，便是需求问题。

而在Standish Group 的 CHAOS 系列报告中，他们将软件项目分为三类——

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)。
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)

在他们于1995年发布的调查报告中。1994年的美国365家公司的8300个项目中，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%。

而在其中，成功项目的影响因素指数最大的，是用户参与；影响问题项目的因素最大的是缺少用户输入；影响失败项目的影响因素最大的是不完整的需求说明。

这些数据说明，一个完整的需求说明是项目不会变成失败项目的基础；而用户的积极参与以及他们的数据的输入，是成功项目的基础之一。

但在近二十年中，即便成功项目的比例逐年增加，但问题项目依旧是占据着最大的比例。

1.1软件的模拟特性

在这些导致需求问题的原因当中，一个最为重要的原因是∶未能很好地理解和掌握"应用"型软件的模拟特性以及由此而产生的一系列影响和要求。软件的模拟特性来源于其知识载体的特性∶软件在运行中表现出来的特性、行为应该和应用的现实情况保持一致。只有这样，用户才会通过观察软件的行为来得出相对应的现实问题的答案，也就是表示，软件成功地对现实进行了模拟。

不过，软件对于现实的模拟，也带来了软件功能的冗余。这些冗余，或多或少是用户所不会使用的，且是开发人员对于需求的理解不透彻，处理不全面而导致的。正是这些冗余对后期的软件维护和开发中在用户需求所要求的功能的设计不完备，使得大量的项目成为了问题项目，更甚者，成为了失败项目。

当然，软件对现实世界的"模拟"并不是机械和被动的。在投入使用之后，它也会通过相应的对外接口对其周围环境产生必要的影响，并进一步帮助人们解决现实世界中遇到的问题。只是它必须以准确的现实理解为基础，在现实的制约之下对外施加影响，进而解决问题。

软件可以被分为三种类别：面向专业用户的纯工具型软件、面向普通用户的纯工具型软件和应用型软件。

专业用户通常以软件为中心开展工作，工具软件是他们的主要手段，因此面向专业用户的纯工具型软件的首要成功标准是要具有功能的复杂性和使用的高效性。功能的复杂可以让专业用户在执行任务时具有更大的发挥空间和回旋余地。使用的高效可以帮助专业用户更快、更好地完成任务。以上两点的实现都要以先进的技术为必要条件。该类软件以创新性为主要关注点，技术创新是它们的生存之道。

普通用户利用软件的目的通常仅限于解决一些实际问题，软件仅仅是一种辅助性的手段，因此面向普通用户的纯工具型软件以功能的有用性为首要成功标准，一些过于复杂的功能反而会因其灵活性而丧失一定的实用性，进而受到用户的抵制。普通用户技术能力有限，所以对操作的要求以使用方便为主，在使用方便的前提下追求使用的高效性。实现功能的有用性和使用的方便性，利用常见的可行技术即可，先进技术并非必要条件。有效性是该类软件的主要关注点，能够有效使用即可占有一席之地。

应用型软件在"模拟"现实的基础之上接收用户的请求，协助用户完成任务，它正确工作的基础是具有"模拟"性。"模拟"性具体是指以下几点∶

1. 目的性。软件的目标是直接或间接地满足用户的某些目的或者解决用户的某些问题，软件的功能是据此设立的。
2. 正确性。软件具备的功能能够保证目标的正确实现。
3. 现实可理解性。软件实现其功能的基础、手段和过程是在用户领域内现实可理解的，即软件系统是在理解其现实环境的基础上，通过影响现实的某些环节，或者改变现实各部分的通信方式，最终达成某些目的或者解决某些问题的。

应用型软件一般以普通用户为应用对象，因此也要求具有使用的方便性。实现功能的"模拟"性和使用的方便性也仅要求所用技术具有可行性。和工具型软件不同，应用型软件通常不是通用的，它们是为特定的应用环境定制的，对环境的"模拟"性是其主要的关注点。

不同的评判标准和关注点决定了3类软件在生产中也会有所不同，尤其是在分析阶段具有截然不同的目标∶ 面向专业用户的工具型软件通常在具有一定的观念创新或技术创新后执行功能分析，分析阶段的主要目的是为充分利用创新优势而进行巧妙的功能安排; 面向普通用户的工具型软件进行分析的主要目的是进行方案权衡，寻找一套切实有效的功能配置;应用型软件分析阶段的主要目的是发现人们利用软件的原因（目的），找出需要软件解决的问题，理解应用环境中的领域知识，保证功能的"模拟"性。

在实际工作中，虽然大部分软件开发人员将其主要精力都消耗在应用型软件的生产中，但他们每天接触更多的却是工具型软件。因此，如果开发人员受到的工具型软件相关评判标准、关注点及生产过程的影响过大，就会对应用型软件的"模拟"特性理解不透彻或应用不坚决，进而导致对需求处理阶段重视不足或者在需求阶段轻视领域知识研究，应用型软件的生产就会发生需求问题。

而在实践当中，对应用型软件的"模拟"特性理解不透彻或应用不坚决的问题的确普遍存在。而在进行需求分析时，人们对软件自身特性投入很大精力的同时，对本应投入很大精力的问题背景和应用环境却常常关注不足。同时，每当软件生产面临时间、市场等其他压力时，漠视"模拟"特性的情况就更为严重

1. 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。 

2.1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出的特性。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 具;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。 

**3.2** **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。

但是，当软件以企业为中心时，它的应用范围会包括企业的各个主要职能部门，包括各部门的主要任务和它们之间任务的协同。这样，该组织的部门划分、传统与惯例、规章和约束、行业特性和行业约定、社会地位和社会价值等组织结构文化和社会背景方面的因素就会对需求分析的正确进行产生一定的影响。而且随着应用范围的扩大，涉众会更加广泛，相互之间的利益冲突也会加剧，因此对商业目标和利益协商的处理要求也变得很有必要，忽视这些非技术性因素会导致整个项目的失败。

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，Boehm2001】。这种递增效应也说明了需求问题的高代价性。

3. 需求工程简介

3.1 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，同的人从不同的角度出发，根据各自不同的理解，会得出不同的定义。

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、记录含求并验证其正确性，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。
3. 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。

需求工程为了完成其任务，需要执行一系列的任务，需求工程活动包括需求开发和需求管理两个方面。需求开发是因为需求工程的"需求"特性而存在的，它们是专门用来处理需求的软件技术，包括需求获取、需求分析、需求规格说明和需求验证4个具体的活动。需求管理是因为需求工程的"工程"特性而存在的，它的目的是在需求发活动之后，保证所确定的需求能够在后续的项目活动中有效地发挥作用，保证各种活动的开展都符合需求要求。

带求获取的目的是从项目的战略规划开始建立最初的原始需求。为此，它需要研究系统将来的应用环境，确定系统的涉众，了解现有的问题，建立新系统的目标，获取为支持新系统目标而需要的业务过程细节和具体的用户需求。

需求分析的目的是保证需求的完整性和一致性。它从需求获取阶段输出的原始需求和业务过程细节出发，将目标、功能和约束映射为软件行为，建立系统模型，然后在抽象后的系统模型中进行分析，标识并修复其中的不一致缺陷，发现并弥补遗漏的需求。

需求规格说明的目的是将完整、一致的需求与能够满足需求的软件行为以文档的方式明确地固定下来。在文档中，可以使用非形式化的文本（如自然语言）描述，也可以使用半形式化的图形语言，如统一建模语言（Unified Modeling Language，UML）描述，还可以使用形式化的语言（如Z语言）描述。描述的结果文档是接下来将被提交进行需求验证的软件需求规格说明。

需求验证是需求开发中的最后一个活动。它的首要目的是保证需求及其文档的正确性，即需求正确地反映了用户的真实意图；它的另一个目标是通过检查和修正，保证需求及其文档的完整性和一致性。需求验证之后的需求及其文档应该是得到所有涉众一致同意的软件需求规格说明，它将作为项目规划、设计、测试、用户手册编写等多个其他软件开发阶段的工作基础，对帮助项目开发人员建立共同的前景具有重要作用。

需求管理是对需求开发所建立的需求基线的管理，它在需求基线完成之后正式开始，并在需求工程阶段结束之后继续存在，在设计、测试、实现等后续的软件系统开发中保证需求作用的持续、稳定发挥。它的主要工作是跟踪后续阶段中的需求实现与需求变更情况，确保需求得到正确的理解和实现。

3.2 需求工程与系统工程

在系统化的需求工程出现之后，需求处理在整个系统开发中所处的位置也出现了变化。

传统的需求处理即软件工程的需求阶段，但系统化的需求工程将软件需求开发和系统需求开发结合起来，在系统工程的开始阶段起到重要作用。在20世纪90年代中期之后，系统需求开发又被称为需求工程的早期阶段（early phase）。软件需求开发相应被称为需求工程的后期阶段（late phase）。

计算系统工程通常是指将计算机引入某一现实系统，并用它来改变现x实系统的运作方式，达到一个理想效果的过程。在计算系统工程中，软件通常具有重要的作用，但系统工程中除了含有处理软件的软件工程之外，还包括硬件工程和人力工程。硬件工程为计算机在某一现实系统中的应用提供硬件支持，如网络布局和处理机配置等。人力工程为计算机在某一现实系统中的应用提供人力资源支持，如维护人员培训、系统管理人员培训和用户培训等。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。正确的方式应该是在处理软件所关注的内容之前就先综合考虑和处理所有的系统因素，包括软件、硬件和人力资源。为此，在实现具体的软件工程、硬件工程和人力工程之前，系统工程需要先进行系统需求开发，以获得对系统整体的综合理解。

系统需求开发的主要目的是获得整个系统的期望目标，包含功能特征和非功能特征（如性能要求等）。为此需要判断系统的涉众，采集他们的目标与要求，研究系统的环境，确定系统的约束，并进行一些整体性的需求分析。系统需求开发阶段的需求分析主要是分析系统的成本效率及系统的组织和行政策略，处理互相依赖、冲突、重叠或不一致的涉众需求，检查并弥补需求缺失，检查技术储备、外部系统等环境约束。系统需求开发的结果会写人系统需求规格说明。

系统需求开发阶段获得的需求将被分配到软件工程、硬件工程或人力工程部分。其中硬件工程和人力工程的需求一般比较容易落实，但软件工程的需求还需要进行更加细致的处理，即进行软件需求开发。软件需求开发用来确定系统需求中应该由软件满足的部分，将其映射为软件行为，产生软件需求规格说明。

软件开发是一个工程性的问题，这一点已经被人们广泛接受。软件开发者的任务就是开发一个软件系统，将之应用于现实世界，并通过软件系统和现实世界的交互，影响和改变现实世界。在这个过程中，软件开发者并不是要从物理结构开始针对问题建立一个特定的计算机，而只是描述所需软件系统的特征和行为，然后通过编程在通用计算机上实现，使之表现出之前所描述的特征和行为。因此软件开发是这样一个工程问题∶利用通用的计算机结构构建一个有用的软件系统，来满足人们的某些目的。

但是，作为一名工程师，软件开发者的工作方式却和其他工程领域的工程师不太一样，这些域包括常见的汽车、电子、化学、航空等。最大的区别在于，一个汽车工程师在开始工作之前，不需要花费精力去研究等待他解决的问题是什么。他们的问题总是设计某种特殊类别的汽车，而且该种汽车的设计目标和特性要求都是清晰与明确的。家庭轿车的设计师不需要考虑让轿车飞行或载重15t等性能之外的要求，也不需要考虑轿车是否应该有轮子或驾驶员应该坐在轿车的前面还是后面等性能之内的问题。每种类型汽车所要解决的问题都是固定的，或者是解决市内交通，或者是解决长途客运等等，没有人会用汽车来建造一栋摩天大厦。因此对于这些工程领域中的工程师而言，一方面因其所从事行业的特殊性给他们施加了很大约束，另一方面也给他们提供了很好的工作基础。

由于计算机应用于现实世界的广泛性，所以软件工程师的工作也具有行业上的广泛性。这就要求他们在不同的行业领域里都表现出优秀的工作能力，例如，一个在金融领域软件开发中成绩斐然的工程师也应有能力在医疗领域进行成功的软件开发。这就带来了问题和解决方案的广泛性。但是软件工程师不可能了解所有的领域，所以在软件开发中他们常常要同时面对新的问题和提出新的解决方案。

因为总要面临新的问题，所以软件工程师常常需要将工作中很大的一部分用来定义问题，然后再为其设计一个新的解决方案。定义问题就是需求工程的任务，所以除了一些特殊情况之外，在软件开发中进行需求工程是非常必要的。

人们很早就认识到需求工程的重要性，正如Frederick Brooks【Brooks1987】所说∶开发软件系统最为困难的部分就是准确说明开发什么。最为困难的概念性工作便是编写详细技术需求，这包括所有面向用户、面向机器和其他软件系统的接口。同时，这也是一旦有错最终将会给系统带来极大损害的部分，并且以后再对它进行修改也极为困难。

虽然需求工程的重要性早就被人们所认识，而且实践也一次又一次地证实了这种认识，但在很多情况下人们还是会忽略需求工程的重要性，这种忽略在学生的校园实践项目中体现得尤为明显。

究其原因，是因为有些特定问题的具体特性掩盖了需求工程的重要性。常见的情况有两类：

① 问题广为人知。像电梯调度、图书管理等问题就属于此类。面对此类问题时，即使不采用需求工程的方法，开发人员也可以得到对问题的准确和全面理解，进而开发出符合要求的系统。

② 问题小而简单。它们开发的代价较小，因此修复的代价也较小，即使全部推倒重来也会有太大的影响，因此该类问题可以不采用需求工程的方法。

而以上两类问题在教学过程中常被当作典型示例，所以导致学生忽略了需求工程的重要性。

【Oboler2003】也在调查中发现，学校的实践和科研项目存在不注重工程化方法（当然也包括需求工程方法）的现象，并总结了两个原因∶第一，项目的结果往往仅要求提交一个可运行的原型系统，而不是完善的产品;第二，一旦项目结束，很少存在后续的完善和维护需要。

这些特定的问题会在一定的场合掩盖需求工程的重要性，但该类情况只是软件开发中的极少数，需求工程的重要性仍然需要得到足够的重视。

Brooks的论断在表明需求工程重要性的同时，，也指出了需求工程的困难性，这来源于需求处理过程的复杂性。

【Lamsweerde 2000】认为，需求工程的复杂性体现在以下几个方面。

1. **处理范围广泛**

需求工程连接现实世界和计算机世界，所以它首先要理解现实世界，要描述现实世界的现状与运行规律，既要描述物理的实体，又要反映人类活动的特点，而且并不存在一切内容都浮于表面的现实世界，需求工程师需要去研究、去发现，才能得到需要描述的内容;其次它也需要理解计算机系统，要清楚计算机与软件的构建方式，要能判断某一方法的可行性;最后它还需要将现实世界和计算机世界连接起来，让它们之间产生期望的互动效应，以满足用户的需求。

1. **涉及诸多参与方**

在需求处理的过程中往往涉及很多参与者，他们来自不同的领域，具有不同的背景、技能、知识层次、关注点、期望值和表达方式等，因此他们之间的交流是非常复杂的，而且他们之间还常常会产生利益冲突和观念冲突，这使得需求处理过程更为复杂。常见的参与方包括客户、用户、领域专家、需求工程师、软件开发者和系统维护者等。

1. **处理内容多样**

需求工程处理的知识内容多种多样，既有用户的功能需求和非功能需求，又有软件将来所处的环境及其约束。用户的功能需求往往体现为多个层次，有高度抽象的战略需求，有粗略的任务需求，也有细节的软件操作需求。除了功能需求之外，需求工程还要关注安全性、可用性、性能、灵活性、健壮性和可维护性等非功能需求，而且这些非功能需求往往还会互相冲突。最终的目标 系统并不仅仅是一个软件，因此需求工程还需要关注目标系统的环境及其约束。环境由人、设备和其他软件组成。

1. **处理活动互相交织**

需求工程包括需求获取、需求分析、需求规格说明和需求验证等4个需求开发活动，它们互相衔接、顺序处理。但基于人类认知逐步深入的特性，人们在分析中发现问题和缺陷时，往往还需要回溯到获取活动，以在更广或更深的范围内获取更多或更详细的信息。所以获取与分析在实际执行中往往是一个互相交织、不断迭代的过程。即使是在需求规格说明和需求验证阶段发现问题和缺陷，修复时也往往要回溯到获取活动或分析活动进行再处理，然后重新执行整个活动过程。所以，需求开发的各项活动虽然在理论上具有顺序处理的特性，但在实际执行过程中往往是迭代和互相交织的。

1. **处理结果要求苛刻**

作为需求处理结果的需求规格说明要满足正确性、完整性和一致性等苛刻要求。因为如果需求规格说明中含有某些不足或错误，就可能会为后续的开发活动或最终的软件产品质量带来灾难性的后果。这些不足和错误包括需求不真实、需求不完整、需求之间互相矛盾、需求模棱两可等。除了以上的不足和错误之外，还有一些小的瑕疵也会带来负面的连锁反应（浪费时间、产生新的错误等），这些瑕疵包括噪声信息、不切实际的要求等。

在软件开发的各项活动中，需求工程的任务是连接现实世界与计算机世界，将现实世界的知识内容转化为计算机世界的工作基础，让软件设计、实现、测试等后续的软件开发活动将精力集中在计算机世界中来。需求工程师是负责完成需求工程主要任务的专门人员，所以他负责衔接现实世界和计算机世界，简单说就是涉众与开发者之间的桥梁。

需求工程师的重要性就体现在他的桥梁作用上。如果没有需求工程师的工作，设计师、程序员等开发者就会在深入并准确理解涉众的想法上出现困难，涉众在见到最终的软件之前也无法把握软件是否满足了他们的想法，最终会导致涉众与开发者之间出现大量的沟通不畅与误解，导致项目返工甚至失败。

需求工程师就是后续软件开发者的代理，负责设计软件方案以满足涉众的各项需求;在面对后续软件开发者时，需求工程师就是涉众的代理，准确地将各项需求告知开发者。

虽然自身属于开发者的一个部分，但是因为需求工程师只有在需求开发阶段需要面对涉众，在更多的后续阶段中面对的是软件开发者，所以好的需求工程师更应该扮演好涉众代理的角色，站在涉众的立场想问题，替涉众跟踪和监控软件开发过程，保护涉众的利益。

因为需求工程比较复杂，又关乎项目的成败，所以合格的需求工程师需要具备多方面的知设与技能。

因为要代表开发者为涉众提供软件解决方案，所以需求工程师必须熟练掌握软件开发方迭与技术，保证设计出来的软件解决方案既是可行的又是成本效益比有效的。

需求工程师还要代表涉众把他们的想法准确地告知开发者，所以需求工程师必须要有非常精确的表达能力，尤其是文档化能力。

当然，最关键的是需求工程师要从涉众那里得到他们的想法并转化为开发者需要的知识，所以需求工程师必须有非常好的交流沟通能力以了解涉众的想法，必须要有抽象建模与分析的能力以准确定义涉众的想法。

依照需求工程师需要完成的开发任务，【Al-Ani2006】将需求工程师应该具备的能力做了定义。

在需求工程师需要具备的能力中，软件开发技术能力是需求工程师最为基础的能力，因为他首先是一个软件开发技术人员。但是需求工程具有连接现实世界的特殊性，所以与其他软件开发角色相比，需求工程师对非技术能力（non-technique skill）的要求并不弱于（甚至强于）对技术能力的要求。这些非技术能力又常被称为软技能（soft skill），是指依赖于需求工程师个人素质而非技术方法掌握情况的能力，通常会涉及认知心理学、社会学、语言学和人类学等人文学科知识。

需求工程师需要掌握的重要软技能包括以下几方面。

1. **交流技能**

需求工程师最为需要的软技能是广义的交流沟通能力，包括表述、写作、面谈、团队工作和义交流能力等。

这里的交流技能指狭义的交流能力，是指需求工程师与他人通过交谈完成信息交换的能力是需求工程师获取需求必备的能力，包括以下两个方面。

首先，需求工程师要掌握交谈和提问的技巧。一方面，需求工程师要通过安排适当的话题，让用户的精力集中在主要的关注点上，然后通过问题的逐步展开和提问方式的灵活变换控制谈话过程，引导用户表达出重要的信息;另一方面，需求工程师要能够和不同的个人或小组展开讨论，要能够应对他们不同的交流方式（木讷、固执、盛气凌人等），剔除他们不同背景的附加影响，导出共同的需求。

其次，需求工程师要具有倾听的技巧。需求工程中的交流是双向交流，因此除了表达自己之外，需求工程师还要学会有效地倾听。有效倾听要能够营造好的交流氛围，抓住其他人说的每一句话，从字里行间中找出重要内容。有效倾听还要求需求工程师站在对方的表达习惯上进行理解，避免用个人的方式来过滤用户表达的信息。

1. **观察技能**

观察也是需求工程师获取需求需要具备的能力。通过观察用户的工作环境和工作过程，需求工程师应该能够发现通过谈话及其他方法所无法发现的重要信息，例如系统对环境的依赖情况和用户没有提及的潜在知识等。即使是在和用户的谈话中也可以通过观察得出用户真实的感受和情感反馈，以便更好地引导谈话的过程，得到理想的信息。

1. **抽象分析与问题解决技能**

首先，需求工程师要具有抽象能力。要能够在面对众多信息内容时清晰地把握问题的重点、核心和本质，要能够从用户的描述当中发现"真实"的需求。这是从获取信息中识别需求必需的技能。

其次，需求工程师要具有整合全局的能力。面对大量来自各种来源的杂乱需求信息片段，要具备从混乱和含糊信息中找出知识的耐心、韧性和组织能力，要能够发现和解决其中的遗漏与冲突，得出完整和一致的系统需求，这是需求工程师验证需求和解决问题必需的能力。

最后，需求工程师要具有系统化思想。在和用户的交流中要能够从全局出发，用不同的方式思考问题，要能够综合考虑不同的需求类别、不同的需求抽象层次和不同的可行性方案，要能够保证需求的系统性和整体性，这既是需求工程师获取需求需要的能力，也是需求工程师进行需求验证和解决问题需要的能力。

1. **写作技能**

需求开发提交的主要成果是书面的需求规格说明，它将被用来在客户、管理人员、开发人员等涉众之间传递信息。信息交流是需求规格说明的第一目的，因此需求规格说明应该具有良好的可理解性，这要求需求工程师需要具备良好的文档组织能力，能够有条理地说明复杂的事物需求规格说明在传递信息的过程中，要保证参与各方对其具有共同的理解，因此需求规格说明应该具有正确性和准确性，这要求需求工程师具有良好的语言驾驭能力，能够清晰地表达复杂的概念。

1. **关系协调和团队工作技能**

需求工程有诸多涉众的参与，因此需求工程师要能够做一个有效的中间协调人，引导各方朝着共同的目标和前景前进。尤其是在参与的各方发生利益或观点冲突时，需求工程师更要能够促成相关方的有效协商，以最终达成一个合理的解决方案。

在面对大型系统时，需求开发往往由团队而不是由个人来进行，因此需求工程师还要能够。调团队内部相互之间的工作与人际关系，建立互相信任的团队工作气氛。

在表面上看，需求工程师的工作只是捕获涉众的想法并忠实地转述给开发者，这使得需求、程师似乎不需要创新能力，甚至于应该抵制创新以与涉众保持一致。但事实上，是否具备创新力往往是判断一个需求工程师是否出色的必要条件。

需求工程师的创新表现在两个方面∶

软件系统并不仅仅是模拟现实，还要让现实变得更好。这需要需求工程师以现实为基构思现实中不存在的软件解决方案，这是一种最基本的创新能力，虽然幅度不是那么明显【Maiden 2007】还发现，涉众在描述现实时，往往会受到现实过度细节的约束或者对抽象的概念解释不清，这时就需要需求工程师运用创新能力，剥离细节约束或者丰富抽象概念细节，建立好的软件解决方案。

出色的需求工程师往往还会给出具有飞跃意义上的创新【Robertson 2002】，如最早的搜索引擎产品、电子产品等概念创新。需要认识到的是，这些创新并不是脱离现实随意构思的与够众不同的想法，而是需要需求工程师敏锐地洞察现实才能实现的创新。它的基础是潜在求——涉众自己都没有认识到但事实上非常需要的东西。所以，需求工程师的创新并没有脱鹏涉众，而是更积极地与涉众保持了一致。

需求工程的复杂性使得一个人很难单独完成复杂系统的需求开发工作，所以实践中需求开发往往是团队行为。【Katzenbach 1993】将团队定义为∶为了一致的目的、绩效目标、方法而共担责任并且技能互补的少数人。

4．对应有内容的建议

在阅读了一到三的内容后，我对这学科有了初步的认识，也了解到了或许以后要做的一些事项。每一个系统、每一个软件的出现，或多或少都经过了这些流程。也许用到的是很让人不爽的软件或者系统，但他对有些用户或客户的需求的分析也是一个可以借鉴的。

在所有东西都逐渐工厂化，流水线化的发展阶段，用着人工的手法进行客户或公司、团队项目的需求的分析，对自己的工作进行目标的指明，或许将会持续很久。没错，在后续，遇到难缠的需求以及复杂的系统，这门课的内容依旧是能用的。即便是再多了几个经典的案例以及调查。

我阅读的时候也发觉，这书关于相关内容的实例的介绍，依旧有点少，更多的像是一本复杂的工具书，而不是教科书。啃起来多多少少有些涩，不太能理解或者没讲透的地方也有不少。

看是否会发生修改吧。