

如何进行软件需求分析

作者：曹伟

需求的定义包括从用户角度（系统的外部行为），以及从开发者角度（一些内部特性）来阐述需求。

关键的问题是一定要编写需求文档。我曾经目睹过一个项目中途更换了所有的开发者，客户被迫与新的需求分析者坐到一起。系统的分析人员说：“我们想与你谈谈你的需求。”客户的第一反应便是：“我已经将我的要求都告诉你们前任了，现在我要的就是给我编一个系统”。而实际上，需求并未编写成文档，因此新的分析人员不得不从头做起。所以如果只有一堆邮件、会谈记录或一些零碎的未整理的对话，你就确信你已明白用户的需求，那完全是自欺欺人。

需求的另外一种定义认为需求是“用户所需要的并能触发一个程序或系统开发工作的说明”。有些需求分析专家拓展了这个概念：“从系统外部能发现系统所具有的满足于用户的特点、功能及属性等”。这些定义强调的是产品是什么样的，而并非产品是怎样设计、构造的。而下面的定义则从用户需要进一步转移到了系统特性：

需求是指明必须实现什么的规格说明。它描述了系统的行为、特性或属性，是在开发过程中对系统的约束。

从上面这些不同形式的定义不难发现：并没有一个清晰、毫无二义性的“需求”术语存在，真正的“需求”实际上在人们的脑海中，这个人们主要是指客户，但一般情况下，用户并不能描述自己的需要，只就需要系统分析人员根据用户的自己语言的描述整理出相关的需要再进一步和客户核对。系统分析员和客户需要确保所有项目风险承担者在描述需求的那些名词的理解上务必达成共识。

任何文档形式的需求（例如如下将要描述的需求规格说明书）仅是一个模型，一种描述。

2. 需求分析的任务

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出详细技术需求，这包括所有面向用户、面向机器和其它软件系统的接口。同时这也是一旦做错，将最终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。

目前，国内产品的庞杂，一家企业可能有几个系统并立运行，它们之间接口是系统开发人员最头痛的问题。

对于商业最终用户应用程序，企业信息系统和软件作为一个大系统的一部分的产品是显而易见的。但是对于我们开发人员来说，并没有编写出客户认可的需求文档，我们如何知道项目于何时结束？而如

如果我们不知道什么对客户来说是重要的，那我们又如何能使客户感到满意呢？

然而，即便并非出于商业目的的软件需求也是必须的。例如库、组件和工具这些供开发小组内部使用的软件。当然你可能偶尔无需文档说明就能与其他人意见较为一致，但更常见的是出现重复返工这种不可避免的后果，而重新编制代码的代价远远超过重写一份需求文档的代价，这些血的教训正在国内的软件开发者身上发生。

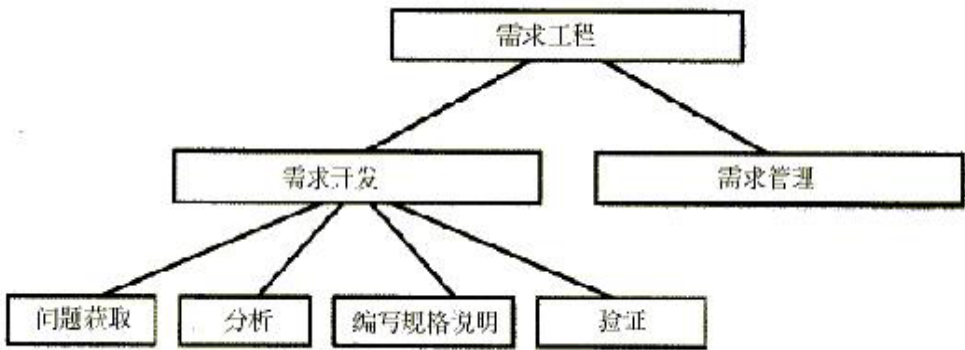
近来，我遇到一个开发小组开发包括代码编辑器在内的一套内部使用的计算机辅助软件。不幸的是，当他们开发完这个工具后，发现这个工具不能打印出源代码文件，使用者当然希望有这个功能。结果这个小组只好手工抄写源代码文档以供代码检查。这说明那怕需求明确无误并构思准确，如果我们没有编写文档，软件达不到期望目标也只能是咎由自取了。

相反的情况，我曾见一个要集成到“错误跟踪系统”中的简单界面写了一页需求说明。而操作系统系统管理员在为处理脚本时发现简单的一张需求清单竟是如此有用。他们依据需求对系统进行测试时，此系统不仅非常清晰地实现了所有必需功能，而且未发现任何错误。

事实上，需求文档在开发过程中一直起指导作用。

3 . 需求分析过程

可把整个软件需求工程研究领域划分为需求开发和需求管理两部分更合适，如图 4-1 所示：



图

4-1 需求工程域的层次分解示意图

需求开发可进一步分为：问题获取、分析、编写规格说明和验证四个阶段。这些子项包括软件类产品中需求收集、评价、编写文档等所有活动。需求开发活动包括以下几个方面：

- 确定产品所期望的用户类别。
- 获取每个用户类的需求。
- 了解实际用户任务和目标以及这些任务所支持的业务需求。

分析源于用户的信息以区别用户任务需求、功能需求、业务规则、质量属性、建议解决方法和附加信息。

将系统级的需求分为几个子系统，并将需求中的一部份分配给软件组件。

了解相关质量属性的重要性。

商讨实施优先级的划分。

将所收集的用户需求编写成文档和模型。

评审需求规格说明，确保对用户需求达到共同的理解与认识，并在整个开发小组接受说明之前将问题都弄清楚。

需求管理需要“建立并维护在软件工程中同客户达成的合同”。这种合同都包含在编写的需求文档与模型中。客户的接受仅是需求成功的一半，开发人员也必须能够接受他们，并真正把需求应用到产品中。通常的需求管理活动包括：

定义需求基线（迅速制定需求文档的主体）。

评审提出的需求变更、评估每项变更的可能影响从而决定是否实施它。

以一种可控制的方式将需求变更融入到项目中。

使当前的项目计划与需求一致。

估计变更需求所产生影响并在此基础上协商新的承诺，这种承诺具体体现在项目解决方案上。

让每项需求都能与其对应的设计、源代码和测试用例联系起来以实现跟踪。

在整个项目过程中跟踪需求状态及其变更情况。

以上几点说明是我总结了成功实施项目后系统分析人员的经验，同时也根据国内外的其他系统实施的相关成功经验，进行了总结。

4．需求的类型

下面这些定义是需求工程领域中常见术语的定义。

软件需求包括三个不同的层次：业务需求、用户需求和功能需求（也包括非功能需求）。

1．业务需求（business requirement）反映了组织机构或客户对系统、产品高层次的目标要求，它们在项目视图与范围文档中予以说明。

2．用户需求(user requirement) 文档描述了用户使用产品必须要完成的任务，这在使用实例（use case）文档或方案脚本说明中予以说明。

3．功能需求(functional requirement)定义了开发人员必须实现的软件功能，使得用户能完成他们的

任务，从而满足了业务需求。

在软件需求规格说明书（SRS）中说明的功能需求充分描述了软件系统所应具有的外部行为。软件需求规格说明在开发、测试、质量保证、项目管理以及相关项目功能中都起了重要的作用。对一个大型系统来说，软件功能需求也许只是系统需求的一个子集，因为另外一些可能属于子系统（或软件部件）。

作为功能需求的补充，软件需求规格说明还应包括非功能需求，它描述了系统展现给用户的行为和执行的操作等。它包括产品必须遵从的标准、规范和合约；外部界面的具体细节；性能要求；设计或实现的约束条件及质量属性。所谓约束是指对开发人员在软件产品设计和构造上的限制。质量属性是通过多种角度对产品的特点进行描述，从而反映产品功能。多角度描述产品对用户和开发人员都极为重要。

下面以一个字处理程序为例来说明需求的不同种类。业务需求可能是：“用户能有效地纠正文档中的拼写错误”，该产品的包装盒封面上可能会标明这是个满足业务需求的拼写检查器。而对应的用户需求可能是“找出文档中的拼写错误并通过一个提供的替换项列表来供选择替换拼错的词”。同时，该拼写检查器还有许多功能需求，如找到并高亮度提示错词的操作；显示提供替换词的对话框以及实现整个文档范围的替换。

从以上定义可以发现，需求并未包括设计细节、实现细节、项目计划信息或测试信息。需求与这些没有关系，它关注的是充分说明你究竟想开发什么。项目也有其它方面的需求，如开发环境需求或发布产品及移植到支撑环境的需求。尽管这些需求对项目成功也至关重要，但它们并非本书所要讨论的。

5．需求分析的原则

不重视需求过程的项目队伍将自食其果。需求工程中的缺陷将给项目成功带来极大风险，这里的“成功”是指推出的产品能以合理的价格、及时地在功能、质量上完全满足用户的期望。下面将讨论一些需求风险。

不适当的需求过程所引起的一些风险：

1. 无足够用户参与

客户经常不明白为什么收集需求和确保需求质量需花费那么多功夫，开发人员可能也不重视用户的参与。究其原因：一是因为开发人员感觉与用户合作不如编写代码有意思；二是因为开发人员觉得已经明白用户的需求了。在某些情况下，与实际使用产品的用户直接接触很困难，而客户也不太明白自己的真正需求。但还是应让具有代表性的用户在项目早期直接参与到开发队伍中，并一同经历整个开

发过程。

系统人员在实践过程中，也有些感觉，在实施一家公司的项目时，若无足够的用户参与，系统人员获得的需求是片面的，不完整的，这样系统在需求之初就埋下风险。

2. 用户需求的不断增加

在开发中若不断地补充需求，项目就越变越庞大以致超过其计划及预算范围。计划并不总是与项目需求规模与复杂性、风险、开发生产率及需求变更实际情况相一致，这使得问题更难解决。实际上，问题根源在于用户需求的改变和开发者对新需求所作的修改。

要想把需求变更范围控制到最小，必须一开始就对项目视图、范围、目标、约束限制和成功标准给予明确说明，并将此说明作为评价需求变更和新特性的参照框架。说明中包括了对每种变更进行变更影响因素分析的变更控制过程，有助于所有风险承担者明白业务决策的合理性，即为何进行某些变更，相应消耗的时间、资源或特性上的折中。

产品开发中不断延续的变更会使其整体结构日渐紊乱，补丁代码也使得整个程序难以理解和维护。插入补丁代码使模块违背强内聚、松耦合的设计原则，特别是如果项目配置管理工作不完善的话，收回变更和删除特性会带来问题。如果你尽早地区别这些可能带来变更的特性，你就能开发一个更为健壮的结构，并能更好地适应它。这样设计阶段需求变更不会直接导致补丁代码，同时也有利于减少因变更导致质量的下降。

3. 模棱两可的需求

模棱两可是需求规格说明中最为可怕的问题。它的一层含义是指诸多读者对需求说明产生了不同的理解；另一层含义是指单个读者能用不止一个方式来解释某个需求说明。

模棱两可的需求会使不同的风险承担者产生不同的期望，它会使开发人员为错误问题而浪费时间，并且使测试者与开发者所期望的不一致。一位系统测试人员曾告诉我，她所在的测试组经常对需求理解有误，以致不得不重写许多测试用例并重做许多测试。

处理模棱两可需求的一种方法是组织好负责从不同角度审查需求的队伍。仅仅简单浏览一下需求文档是不能解决模棱两可问题的。如果不同的评审者从不同的角度对需求说明给予解释，但每个评审人员都真正了解需求文档，这样二义性就不会直到项目后期才被发现，那时再发现的话会使得更正代价很大。

4. 不必要的特性

“画蛇添足”是指开发人员力图增加一些“用户欣赏”但需求规格说明中并未涉及的新功能。经常发生

的情况是用户并不认为这些功能性很有用，以致在其上耗费的努力“白搭”了。开发人员应当为客户构思方案并为他们提供一些具有创新意识的思路，具体提供哪些功能要在客户所需与开发人员在允许时限内的技术可行性之间求得平衡，开发人员应努力使功能简单易用，而不要未经客户同意，擅自脱离客户要求，自作主张。

同样，客户有时也可能要求一些看上去很“酷”，但缺乏实用价值的功能，而实现这些功能只能徒耗时间和成本。为了将“画蛇添足”的危害尽量减小，应确信：你明白为什么要包括这些功能，以及这些功能的“来龙去脉”，这样使得需求分析过程始终是注重那些能使用户完成他们业务任务的核心功能。

5. 过于精简的规格说明

有时，客户并不明白需求分析有如此重要，于是只作一份简略之至的规格说明，仅涉及了产品概念上的内容，然后让开发人员在项目进展中去完善，结果很可能出现的是开发人员先建立产品的结构之后再完成需求说明。这种方法可能适合于尖端研究性的产品或需求本身就十分灵活的情况。但在大多数情况下，这会给开发人员带来挫折（使他们在不正确的假设前提和极其有限的指导下工作），也会给客户带来烦恼（他们无法得到他们所设想的产品）。

6. 忽略了用户分类

大多数产品是由不同的人使用其不同的特性，使用频繁程度也有所差异，使用者受教育程度和经验水平也不尽相同。如果你不能在项目早期就针对所有这些主要用户进行分类的话，必然导致有的用户对产品感到失望。例如，菜单驱动操作对高级用户太低效了，但含义不清的命令和快捷键又会使不熟练的用户感到困难。

7. 不准确的计划

据统计，导致需求过程中软件成本估计极不准确的原因主要有以下五点：频繁的需求变更、遗漏的需求、与用户交流不够、质量低下的需求规格说明和不完善的需求分析。

对不准确的要求所提问题的正确响应是“等我真正明白你的需求时，我就会来告诉你”。基于不充分信息和未经深思的对需求不成熟的估计很容易为一些因素左右。要作出估计时，最好还是给出一个范围。未经准备的估计通常是作为一种猜测给出的，听者却认为是一种承诺。因此我们要尽力给出可达到的目标并坚持完成它。

6. 需求分析人员和用户的合作关系

优秀的软件产品是建立在优秀的需求基础之上的。而高质量的需求来源于客户与开发人员之间有效的交流与合作。通常，开发人员与客户或客户代理人，如市场人员间的关系反而会成为一种对立关系。

双方的管理者都只想自己的利益而搁置用户提供的需求从而产生摩擦，在这种情况下，不会给双方带来一点益处。

只有当双方参与者都明白要成功自己需要什么，同时也应知道要成功合作方需要什么时，才能建立起一种合作关系。由于项目压力与日渐增，所有风险承担者有着一个共同的目标这一点容易被遗忘。其实大家都想开发出一个既能实现商业价值，又能满足用户需要，还能使开发者感到满足的优秀软件产品。

软件客户需求权利书列出了十条关于客户在项目需求工程实施中与分析人员、开发人员交流时的合法要求。每一项权利都对应着软件开发人员、分析人员的义务。而软件客户需求义务书也列出了十条关于客户在需求过程中应承担的义务。如果愿意，可以将其作为开发人员的权利书。

客户有如下权利：

1：要求分析人员使用符合客户语言习惯的表达

需求讨论应集中于业务需要和任务，故要使用业务术语，你应将其教给分析人员，而你 不一定要懂得计算机的行业术语。

2：要求分析人员了解客户的业务及目标

通过与用户交流来获取用户需求、分析人员才能更好地了解你的业务任务和怎样才能使产品更好地满足你的需要。这将有助于开发人员设计出真正满足你的需要并达到你期望的优秀软件。为帮助开发人员和分析人员，可以考虑邀请他们观察你或你的同事是怎样工作的。如果新开发系统是用来替代已有的系统，那么开发人员应使用一下目前的系统，这将有利于他们明白目前系统是怎样工作的，其工作流程的情况，以及可供改进之处。

3：要求分析人员编写软件需求规格说明

分析人员要把从你和其他客户那里获得的所有信息进行整理，以区分开业务需求及规范、功能需求、质量目标、解决方法和其它信息。通过这些分析就能得到一份软件需求规格说明。而这份软件需求规格说明便在开发人员和客户之间针对要开发的产品内容达成了协议。软件需求规格说明书可以用一种你认为易于翻阅和理解的方式组织编写。要评审编写出的规格说明以确保它们准确而完整地表达了你的需求。一份高质量的软件需求规格说明能有助于开发人员开发出真正需要的产品。

4：要求得到需求工作结果的解释说明

分析人员可能采用了多种图表作为文字性软件需求规格说明的补充。因为如工作流程图那样的图表能很清楚地描述出系统行为的某些方面。所以需求说明中的各种图表有着极高的价值。虽然它们不太

难于理解，但是你很可能对此并不熟悉。因此可以要求分析人员解释说明每张图表的作用或其它的需求开发工作结果和符号的意义，及怎样检查图表有无错误及不一致等。

5：要求开发人员尊重你的意见

如果用户与开发人员之间不能相互理解，那关于需求的讨论将会有障碍，共同合作能使大家“兼听则明”。参与需求开发过程的客户有权要求开发人员尊重他们并珍惜他们为项目成功所付出的时间。同样，客户也应对开发人员为项目成功这一共同目标所作出的努力表示尊重与感激。

6：要求开发人员对需求及产品实施提供建议，拿出主意

通常，客户所说的“需求”已是一种实际可能的实施解决方案，分析人员将尽力从这些解决方法中了解真正的业务及其需求，同时还应找出已有系统不适合当前业务之处，以确保产品不会无效或低效。在彻底弄清业务领域内的事情后，分析人员有时就能提出相当好的改进方法。有经验且富有创造力的分析人员还能提出增加一些用户并未发现的很有价值的系统特性。

7：描述产品易使用的特性

你可以要求分析人员在实现功能需求的同时还要注重软件的易用性。因为这些易用特性或质量属性能使你更准确、高效地完成任务。例如，客户有时要求产品要“用户友好”或“健壮”或“高效率”，但对于开发人员来说，太主观了并无实用价值。正确的应是：分析人员通过询问和调查了解客户所要的友好、健壮、高效所包含的具体特性。

8：调整需求，允许重用已有的软件组件

需求通常要有一定的灵活性。分析人员可能发现已有的某个软件组件与你描述的需求很相符。在这种情况下，分析人员应提供一些修改需求的选择以便开发人员能够在新系统开发中重用一些已有的软件。如果有可重用的机会出现，同时你又能调整你的需求说明，那就能降低成本和节省时间，而不必严格按原有的需求说明开发。所以说，如果想在产品中使用一些已有的商业常用组件，而它们并不完全适合你所需的特性，这时一定程度上的需求灵活性就显得极为重要了。

9：获得满足客户功能和质量要求的系统

每个人都希望项目获得成功。但这不仅要求你要清晰地告知开发人员关于系统“做什么”所需的所有信息，而且还要求开发人员能通过交流了解清楚取舍与限制。一定要明确说明你的假设和潜在的期望。否则，开发人员开发出的产品很可能无法让你满意。

客户有下列义务：

1：给分析人员讲解你的业务

分析人员要依靠你给他们讲解的业务概念及术语。但你不能指望分析人员会成为该领域的专家，而只能让他们真正明白你的问题和目标。不要期望分析人员能把握你们业务的细微与潜在之处，他们很可能并不知道那些对于你和你的同事来说理所当然的“常识”。

2：抽出时间清楚地说明并完善需求

客户很忙，经常在最忙的时候还得参与需求开发。但无论如何，你有义务抽出时间参与“头脑风暴”会议的讨论，接受采访或其它获取需求的活动。有时分析人员可能先以为明白了你的观点，而过后发现还需要你的讲解。这时，请耐心一些对待需求和需求的精化工作过程中的反复，因为它是人们交流中的很自然的现象，何况这对软件产品的成功极为重要。

3：准确而详细地说明需求

编写一份清晰、准确的需求文档是很困难的。由于处理细节问题不但烦人而且又耗时，故很容易留下模糊不清的需求。但是，在开发过程中，必须得解决这种模糊性和不准确性。而你恰是为解决这些问题作出决定的最佳人选。不然的话，你就只好靠开发人员去正确猜测了。在需求规格说明中暂时加上待定（to be determined, TBD 也可采用汉语拼音略写“DQD：待确定”）的标志是个不错的办法。用该标志可指明了哪些需要进一步探讨、分析或增加信息的地方。不过，有时也可能因为某个特殊需求难以解决或没有人愿意处理它而注上 TBD 标志。尽量将每项需求的内容都阐述清楚，以便分析人员能准确的将其写进软件需求规格说明中。如果你一时不能准确表述，那就得允许获取必要的准确信息这样一个过程。通常使用所谓的原型技术。通过开发的原型，你可以同开发人员一起反复修改，不断完善需求定义。

4：及时地作出决定

正如一位建筑师为你修建房屋，分析人员将要求你做出一些选择和决定。这些决定包括来自多个用户提出的处理方法或在质量特性冲突和信息准确度中选择折衷方案等。有权做出决定的客户必须积极地对待这一切，尽快做处理、做决定。因为开发人员通常只有等你做出了决定才能行动，而这种等待会延误项目的进展。

5：尊重开发人员的需求可行性及成本评估

所有的软件功能都有其成本价格，开发人员最适合预算这些成本（尽管许多开发人员并不擅长评估预测）。你所希望的某些产品特性可能在技术上行不通，或者实现它要付出极为高昂的代价。而某些需求试图在操作环境中要求不可能达到的性能或试图得到一些根本得不到的数据，开发人员会对此作出负面的评价意见，你应该尊重他们的意见。有时，你可以重新给出一个在技术上可行、实现上便宜

的需求，例如，要求某个行为在“瞬间”发生是不可行的，但换种更具体的时间需求说法（“在 50ms 以内”，但若没有准确的技术分析不能轻易下结论），这就可以实现了。

6：划分需求优先级别

大多数项目没有足够的时间或资源来实现功能性的每个细节。决定哪些特性是必要的，哪些是重要的，哪些是好的，是需求开发的主要部分。只能由你来负责设定需求优先级，因为开发者并不可能按你的观点决定需求优先级。开发者将为你确定优先级提供有关每个需求的花费和风险的信息。当你设定优先级时，你帮助开发者确保在适当的时间内用最小的开支取得最好的效果。在时间和资源限制下，关于所需特性能否完成或完成多少应该尊重开发人员的意见。尽管没有人愿意看到自己所希望的需求在项目中未被实现，但毕竟是要面对这种现实的。业务决策有时不得不依据优先级来缩小项目范围或延长工期，或增加资源，或在质量上寻找折衷。

7：评审需求文档和原型

正如我们将在第 14 章讨论的，无论是正式的还是非正式的方式，对需求文档进行评审都会对软件质量提高有所帮助。让客户参与评审才能真正鉴别需求文档是否的确完整、正确说明了期望的必要特性。评审也给客户代表提供一个机会，给需求分析人员带来反馈信息以改进他们的工作。如果你认为编写的需求文档不够准确，就有义务尽早告诉分析人员并为改进提供建议。通过阅读需求规格说明，很难想象实际的软件是什么样子的。更好的方法是先为产品开发一个原型。这样你就能提供更有价值的反馈信息给开发人员，帮助他们更好地理解你的需求。必须认识到：原型并非是一个实际产品，但开发人员能将其转变、扩充成功能齐全的系统。

8：需求出现变更要马上联系

不断的需求变更会给在预定计划内完成高质量产品带来严重的负面影响。变更是不可避免的，但在开发周期中变更越在晚期出现，其影响越大。变更不仅会导致代价极高的返工，而且工期也会被迫延误，特别是在大体结构已完成后又需要增加新特性时。所以一旦你发现需要变更需求时，请一定立即通知分析人员。

9：应遵照开发组织处理需求变更的过程

为了将变更带来的负面影响减少到最低限度，所有的参与者必须遵照项目的变更控制过程。这要求不放弃所有提出的变更，对每项要求的变更进行分析、综合考虑，最后作出合适的决策以确定将某些变更引入项目中。

10：尊重开发人员采用的需求工程过程

软件开发中最具挑战性的莫过于收集需求并确定其正确性。分析人员采用的方法有其合理性。也许你认为需求过程不太划算，但请相信花在需求开发上的时间是“很有价值”的。如果你理解并支持分析人员为收集、编写需求文档和确保其质量所采用的技术，那么整个过程将会更为顺利。尽管去询问分析人员为什么他们要收集某些信息，或参与与需求有关的活动。

系统分析人员在开发过程中可能会遇到以下问题，一些很忙的客户可能不愿意积极参与需求过程，而缺少客户参与将很可能导致不理想的产品。故一定要确保需求开发中的主要参与者都了解并接受他们的义务。如果遇到分歧，通过协商以达成对各自义务的相互理解，这样能减少今后的摩擦。

7. 需求文档

需求开发的最终成果是：客户和开发小组对将要开发的产品达成一致协议。协议综合了业务需求、用户需求和软件功能需求。就像我们早先所看到的，项目视图和范围文档包含了业务需求，而使用实例文档则包含了用户需求。你必须编写从使用实例派生出的功能需求文档，还要编写产品的非功能需求文档，包括质量属性和外部接口需求。只有以结构化和可读性方式编写这些文档，并由项目的风险承担者评审通过后，各方面人员才能确信他们所赞同的需求是可靠的。

你可以使用以下三种方法编写软件需求规格说明：

用好的结构化和自然语言编写文本型文档。

建立图形化模型，这些模型可以描绘转换过程、系统状态和它们之间的变化、数据关系、逻辑流或对象类和它们的关系。

编写形式化规格说明，这可以通过使用数学上精确的形式化逻辑语言来定义需求。

由于形式化规格说明具有很强的严密性和精确度，因此，所使用的形式化语言只有极少数软件开发人员才熟悉，更不用说客户了。虽然结构化的自然语言具有许多缺点，但在大多数软件工程中，它仍是编写需求文档最现实的方法。包含了功能和非功能需求的基于文本的软件需求规格说明已经为大多数项目所接受。图形化分析模型通过提供另一种需求视图，增强了软件需求规格说明。

软件需求规格说明不仅是系统测试和用户文档的基础，也是所有子系列项目规划、设计和编码的基础。它应该尽可能完整地描述系统预期的外部行为 and 用户可视化行为。除了设计和实现上的限制，软件需求规格说明不应该包括设计、构造、测试或工程管理的细节。许多读者使用软件需求规格说明来达到不同的目的：

客户和营销部门依赖它来了解他们所能提供的产品。

项目经理根据包含在软件需求规格说明中描述的产品来制定规划并预测进度安排、工作量和资源。

软件开发小组依赖它来理解他们将要开发的产品。

测试小组使用软件需求规格说明中对产品行为的描述制定测试计划、测试用例和测试过程。

软件维护和支持人员根据需求规格说明了解产品的某部分是做什么的。

产品发布组在需求规格说明和用户界面设计的基础上编写客户文档，如用户手册和帮助屏幕等。

培训人员根据需求规格说明和用户文档编写培训材料。

软件需求规格说明作为产品需求的最终成果必须具有综合性：必须包括所有的需求。开发者和客户不能作任何假设。如果任何所期望的功能或非功能需求未写入软件需求规格说明，那么它将不能作为协议的一部分并且不能在产品中出现。

我见过有一个项目突然接到测试人员发出的错误灾难的报告。结果是他们测试的是老版本的软件需求规格说明，而他们觉得错误的地方正是产品所独有的特性。他们的测试工作是徒劳的，因为他们一直在老版本的软件需求规格说明中寻找错误的系统行为。

在编写软件需求规格说明，希望读者牢记以下的建议：

对节、小节和单个需求的号码编排必须一致。

在右边部分留下文本注释区。

允许不加限制地使用空格。

正确使用各种可视化强调标志（例如，黑体、下划线、斜体和其它不同字体）。

创建目录表和索引表有助于读者寻找所需的信息。

对所有图和表指定号码和标识号，并且可按号码进行查阅。

使用字处理程序中交叉引用的功能来查阅文档中其它项或位置，而不是通过页码或节号。

为了满足软件需求规格说明的可跟踪性和可修改性的质量标准，必须唯一确定每个软件需求。这可以使你在变更请求、修改历史记录、交叉引用或需求的可跟踪矩阵中查阅特定的需求。由于要达到这一目的，用单一的项目列表是不够的，因此，我将描述几个不同的需求标识方法，并阐明它们的优点与缺点。可以选择最适合你的方法。

(1) 序列号最简单的方法是赋予每个需求一个唯一的序列号，例如 SRS-13。当一个新的需求加入到商业需求管理工具的数据库之后，这些管理工具就会为其分配一个序列号（许多这样的工具也支持层次化编号）。序列号的前缀代表了需求类型，例如 SRS 代表“软件需求说明”。由于序列号不能重用，所以把需求从数据库中删除时，并不释放其所占据的序列号，而新的需求只能得到下一个可用的序列号。这种简单的编号方法并不能提供任何相关需求在逻辑上或层次上的区别，而且需求的标识不能提

供任何有关每个需求内容的信息。

(2) 层次化编码这也许是最常用的方法。如果功能需求出现在软件需求规格说明中第 3.2 部分,那么它们将具有诸如 3.2.4.3 这样的标识号。标识号中的数字越多则表示该需求越详细,属于较低层次上的需求。即使在一个中型的软件需求规格说明中,这些标识号也会扩展到许多位数字,并且这些标识号也不提供任何有关每个需求目的的信息。如果你要插入一个新的需求,那么该需求所在部分其后所有需求的序号将要增加。删除或移去一个需求,那么该需求所在部分其后所有需求的序号将要减少。但其他地方的引用将混乱,对于这种简单的层次化编号的一种改进方法是对需求中主要的部分进行层次化编号,然后对于每个部分中的单一功能需求用一个简短文字代码加上一个序列号来识别。例如,软件需求规格说明可能包含“第 3.2.5 部分—编辑功能”,并将此部分编写成子模块文档,然后配置管理。

有时,你觉得缺少特定需求的某些信息。在解决这个不确定性之前,可能必须与客户商议、检查与另一个系统的接口或者定义另一个需求。使用“待确定”(to be determined, TBD 或采用汉语拼音略写 DQD)符号作为标准指示器来强调软件需求规格说明中这些需求的缺陷。通过这种方法,你可以在软件需求规格说明中查找所要澄清需求的部分。记录谁将解决哪个问题、怎样解决及什么时候解决。把每个 TBD 编号并创建一个 TBD 列表,这有助于方便地跟踪每个项目。

在继续进行构造需求集合之前,必须解决所有的 TBD 问题,因为任何遗留下来的不确定问题将会增加出错的风险和需求返工。当开发人员遇到一个 TBD 问题或其它模糊之处时,他可能不会返回到原始需求来解决问题。多半开发者对它进行猜测,但并不总是正确的。如果有 TBD 问题尚未解决,而你又要继续进行开发工作,那么尽可能推迟实现这些需求,或者解决这些需求的开放式问题,把产品的这部分设计得易于更改。

编写优秀的需求文档没有现成固定的方法,最好是根据经验进行。从过去所遇到的问题中可使你受益匪浅。许多需求文档可以通过使用有效的技术编写风格和使用用户术语而不是计算机专业术语的方式得以改进。

你在编写优秀的需求文档时,希望读者还需牢记以下几点建议:

保持语句和段落的简短。

采用主动语态的表达方式。

编写具有正确的语法、拼写和标点的完整句子。

使用的术语与词汇表中所定义的应该一致。

需求陈述应该具有一致的样式，例如“系统必须..”或者“用户必须..”，并紧跟一个行为动作和可观察的结果。例如，“仓库管理子系统必须显示一张所请求的仓库中有存货的库存清单。”

为了减少不确定性，必须避免模糊的、主观的术语，例如，用户友好、简单、有效、最新技术、优越的、可接受的等。当用户说“用户友好”或者“快”时，你应该明确它们的真正含义并且在需求中阐明用户的意图。

避免使用比较性的词汇，定量地说明所需要提高的程度或者说清一些参数可接受的最大值和最小值。当客户说明系统应该“处理”、“支持”或“管理”某些事情时，你应该能理解客户的意图。由于需求的编写是层次化的，因此，可以把顶层不明确的需求向低层详细分解，直到消除不明确性为止。

文档的编写人员不应该把多个需求集中在一个冗长的叙述段落中。在需求中诸如“和”，“或”之类的连词就表明了该部分集中了多个需求。务必记住，不要在需求说明中使用“和/或”，“等等”之类的连词。

8. 需求分析的过程

需求获取是在问题及其最终解决方案之间架设桥梁的第一步。获取需求的一个必不可少的结果是对项目中描述的客户需求的普遍理解。一旦理解了需求，分析者、开发者和客户就能探索出描述这些需求的多种解决方案。参与需求获取者只有在他们理解了问题之后才能开始设计系统，否则，对需求定义的任何改进，设计上都必须大量的返工。把需求获取集中在用户任务上——而不是集中在用户接口上——有助于防止开发组由于草率处理设计问题而造成的失误。

需求获取、分析、编写需求规格说明和验证并不遵循线性的顺序，这些活动是相互隔开、增量和反复的。当你和客户合作时，你就将会问一些问题，并且取得他们所提供的信息（需求获取）。同时，你将处理这些信息以理解它们，并把它分成不同的类别，还要把客户需求同可能的软件需求相联系。然后，你可以使客户信息结构化，并编写成文档和示意图。下一步，就可以让客户代表评审文档并纠正存在的错误。这四个过程贯穿着需求开发的整个阶段。

由于软件开发项目和组织文化的不同，对于需求开发没有一个简单的、公式化的途径。下面列出了14个步骤，你可以利用它们指导你的需求开发活动。对于需求的任何子集，一旦你完成了第十三步，那么你就可以很有信心地继续进行系统的每一部分的设计、构造，因为你将开发出一个好的产品：

1. 定义项目的视图和范围。
2. 确定用户类。
3. 在每个用户类中确定适当的代表。

4. 确定需求决策者和他们的决策过程。
5. 选择你所用的需求获取技术。
6. 运用需求获取技术对作为系统一部分的使用实例进行开发并设置优先级。
7. 从用户那里收集质量属性的信息和其它非功能需求。
8. 详细拟订使用实例使其融合到必要的功能需求中。
9. 评审使用实例的描述和功能需求。
10. 如果有必要，就要开发分析模型用以澄清需求获取的参与者对需求的理解。
11. 开发并评估用户界面原型以助想像还未理解的需求。
12. 从使用实例中开发出概念测试用例。
13. 用测试用例来论证使用实例、功能需求、分析模型和原型。
14. 在继续进行设计和构造系统每一部分之前，重复 6~13 步。

需求获取可能是软件开发中最困难、最关键、最易出错及最需要交流的方面。需求获取只有通过有效的客户—开发者的合作才能成功。分析者必须建立一个对问题进行彻底探讨的环境，而这些问题与产品有关。为了方便清晰地进行交流，就要列出重要的小组，而不是假想所有的参与者都持有相同的看法。对需求问题的全面考察需要一种技术，利用这种技术不但考虑了问题的功能需求方面，还可讨论项目的非功能需求。确定用户已经理解：对于某些功能的讨论并不意味着即将在产品中实现它。对于想到的需求必须集中处理并设定优先级，以避免一个不能带来任何益处的无限大的项目。

需求获取是一个需要高度合作的活动，而并不是客户所说的需求的简单拷贝。作为一个分析者，你必须透过客户所提出的表面需求理解他们的真正需求。询问一个可扩充的问题有助于你更好地理解用户目前的业务过程并且知道新系统如何帮助或改进他们的工作。

需求获取利用了所有可用的信息来源，这些信息描述了问题域或在软件解决方案中合理的特性。一个研究表明：比起不成功的项目，一个成功的项目在开发者和客户之间采用了更多的交流方式。与单个客户或潜在的用户组一起座谈，对于业务软件包或信息管理系统（MIS）的应用来说是一种传统的需求来源。

在每一次座谈讨论之后，记下所讨论的条目，并请参与讨论的用户评论并更正。及早并经常进行座谈讨论是需求获取成功的一个关键途径，因为只有提供需求的人才能确定是否真正获取需求。进行深入收集和分析以消除任何冲突或不一致性。尽量理解用户用于表述他们需求的思维过程。充分研究用户执行任务时作出决策的过程，并提取出潜在的逻辑关系。流程图和决策树是描述这些逻辑决策途径

的好方法。

当进行需求获取时，应避免受不成熟的细节的影响。在对切合的客户任务取得共识之前，用户能很容易地在一个报表或对话框中列出每一项的精确设计。如果这些细节都作为需求记录下来，他们会给随后的设计过程带来不必要的限制。你可能要周期性地检查需求获取，以确保用户参与者将注意力集中在与今天所讨论的话题适合的抽象层上。向他们保证在开发过程中，将会详尽阐述他们的需求。

在一个逐次详细描述过程中，重复地详述需求，以确定用户目标和任务，并作为使用实例。然后，把任务描述成功能需求，这些功能需求可以使用户完成其任务，也可以把它们描述成非功能需求，这些非功能需求描述了系统的限制和用户对质量的期望。虽然最初的屏幕构思有助于描述你对需求的理解，但是你必须细化用户界面设计。

作者小传：

曹伟，南京易点网络软件公司技术总监。从 ERP 的系统开发，对整体系统有较强的认识欲望，现正在实施企业级软件系统构架，实施一个国际化企业电子化的解决方案。