第9章　编写需求文档

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。这一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。

你可以用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。

本章介绍软件需求规格说明的目的和结构，包括一个建议性的文档模板。同时还提供编写功能需求规格说明的原则并附带讲述几个不完善的需求陈述以及改进建议的例子。在第 10 章将介绍利用图形化技术表示需求。本书并不深入介绍形式化需求方法；若要深入讨论形式化需求方法，可参考 Alan Davis编著的《软件需求：对象、功能和说明》（1993）。

9.1　软件需求规格说明

软件需求规格说明，也称为功能规格说明、需求协议以及系统规格说明。它精确地阐述一个软件系统必须提供的功能和性能以及它所要考虑的限制条件。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和用户可视化行为。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、测试用例和试过程。

- 软件维护和支持人员根据SRS了解产品的某部分是做什么的。
- 产品发布组在SRS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SRS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。

毫无疑问，你必须在开始设计和构造之前编写出整个产品的软件需求规格说明。你可以反复地或以渐增方式编写需求规格说明，这取决于以下几个因素：是否可以一开始就确定所有的需求，编写软件需求规格说明的人是否将参与开发系统，计划发行的版本数量等等。然而，每个项目针对要实现的每个需求集合必须有一个基准协议，基准（ baseline）是指正在开发的软件需求规格说明向已通过评审的软件需求规格说明的过渡过程。必须通过项目中所定义的变更控制过程来更改基准软件需求规格说明。

所有的参与者必须根据已通过评审的需求来安排工作以避免不必要的返工和误解。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。

第1章提出了高质量需求文档所具有的特征：完整性、一致性、可更改性和可跟踪性。构造并编写软件需求规格说明，并使用户和其它读者能理解它（ Sommerville and Sawyer 1997）。牢记以下可读性的建议：

- 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。

9.1.1 标识需求

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 UR-2或SRS13。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 UR代表“ 用户需求 ”。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。这种方法简单且紧凑，你使用的字处理程序可能可以自动编排序号。然而，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。这些变化将破坏系统其它地方需求的引用。

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 ”，那么这一部分需求的编号可以写成ED-1、ED-2等等。这种方法具有层次性和组织性，同时使标识号简短和具有一定意义并与位置无关等特点。如果要在ED-1和ED-2之间插入新的需求ED-9，则不必对该部分中余下的需求重新编号。

3) 层次化文本标签 Tom Gilb顾问提出基于文本的层次化标签方案来标识单个需求（ Gilb 1988）。考虑这样一个需求：“ 当用户请求打印超过 10个副本时，系统必须让用户进行确认判断。” 这一需求可能被标识为 PRINT.COPIES.CONFIRM，这意味着这个需求是打印功能的一部分，并且与要打印的副本数量的设置问题有关。层次化文本标签是结构化的，具有语义上的含义，并且不受增加、删除或移动其它需求的影响。它们的主要缺点是文本标签比层次化数字标识码复杂。

9.1.2 处理不完整性

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）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（gap）。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。

9.1.3 用户界面和软件需求规格说明

把用户界面的设计编入软件需求规格说明既有好处也有坏处。消极方面，屏幕映像和用户界面机制是解决方案（设计）的描述，而不是需求。如果你在完成了用户界面的设计之后才能确定软件需求规格说明，那么需求开发的过程将会花费很长的时间。这将会使那些只关心需求开发时间的经理、客户或开发人员失去耐心。用户界面的布局不能替代定义功能需求。不要指望开发人员可以从屏幕中推断出潜在的功能和数据关系。把用户界面的设计加入软件需求规格说明还意味着开发人员在更改一个用户界面的元素时必须相应更改需求的过程。

积极方面，探索潜在的用户界面有助于你精化需求并使用户—系统的交互对用户和开发人

员更具有实在性。用户界面的演示也有助于项目计划的制定和预测。根据以往类似开发活动的经验，你可以数清图形用户界面（ GUI）的元素数目或者计算与每个屏幕有关的功能点[⊖]数目，然后估计实现这些屏幕功能所需的工作量。

一个合理的权衡点是：在软件需求规格说明中加入所选择的用户界面组件的概念映像草图，而在实现时并不一定要精确地遵循这些方法。通过使用另一种方式来表示需求，从而增强相互交流的能力，但并不增加对开发人员的限制，也不增加变更管理过程的负担。例如，一个复杂对话框的最初草案将描述支持需求部分的意图，但是一个有经验的设计者可以把它转化为一个带有标签组件的对话框或使用其它方法以提高其可用性。

9.2 软件需求规格说明模板

每个软件开发组织都应该在它们的项目中采用一种标准的软件需求规格说明的模板。有许多推荐的软件需求规格说明模板可以使用（ Davis 1993；Robertson and Robertson 1999）。Dorfman和Thayer（1990）从美国国家标准局、美国国防部、美国宇航局以及许多英国和加拿大的有关部门中收集了20几个需求标准和许多实例。许多人使用来自 IEEE标准830-1998的模板，“ IEEE推荐的软件需求规格说明的方法 ”（IEEE 1998）。这是一个结构好并适用于许多种软件项目的灵活的模板。

图9-1描绘了从IEEE 830标准改写并扩充的软件需求规格说明的模板。可以根据项目的需要来修改这个模板。如果模板中某一特定部分不适合你的项目，那么就在原处保留标题，并注明该项不适用。这将防止读者认为是否不小心遗漏了一些重要的部分。与其它任何软件项目文档一样，该模板包括一个内容列表和一个修正的历史记录，该记录包括对软件需求规格说明所作的修改、修改日期、修改人员和修改原因。

- a. **引言**
  - a.1 目的
  - a.2 文档约定
  - a.3 预期的读者和阅读建议
  - a.4 产品的范围
  - a.5 参考文献
- b. **综合描述**
  - b.1 产品的前景
  - b.2 产品的功能
  - b.3 用户类和特征
  - b.4 运行环境
  - b.5 设计和实现上的限制
  - b.6 假设和依赖
- c. **外部接口需求**
  - c.1 用户界面
  - c.2 硬件接口
  - c.3 软件接口
  - c.4 通信接口
- d. **系统特性**
  - d.1 说明和优先级
  - d.2 激励/响应序列
  - d.3 功能需求
- e. **其它非功能需求**
  - e.1 性能需求
  - e.2 安全设施需求
  - e.3 安全性需求
  - e.4软件质量属性
  - e.5业务规则
  - e.6 用户文档
- f. **其它需求**

附录A：词汇表

附录B：分析模型

附录C：待确定问题的列表

图9-1 软件需求规格说明模板

⊖ 功能点是对一个应用程序中用户可见功能的数量的测量，而与如何构造功能无关。你可以根据内部逻辑文件、外部接口文件以及外部输入、输出和请求的数量，从对用户需求的理解中估计功能点（IFPUG 1999）。

本章的剩余部分将描述图 9-1所示的模板中每一部分应包含的信息。你可以通过参考其它已编写好的项目文档（例如项目视图和范围文档或接口规格说明）来将每一部分内容具体化，而不是复制信息或者把所有的内容组成一个单一的文档。不要生搬硬套这个模板，应该把这个模板转换为你所需要的文档。

a. 引言

引言提出了对软件需求规格说明的纵览，这有助于读者理解文档如何编写并且如何阅读和解释。

a.1目的

对产品进行定义，在该文档中详尽说明了这个产品的软件需求，包括修正或发行版本号。如果这个软件需求规格说明只与整个系统的一部分有关系，那么就只定义文档中说明的部分或子系统。

a.2 文档约定

描述编写文档时所采用的标准或排版约定，包括正文风格、提示区或重要符号。例如，说明了高层需求的优先级是否可以被其所有细化的需求所继承，或者每个需求陈述是否都有其自身的优先级。

a.3 预期的读者和阅读建议

列举了软件需求规格说明所针对的不同读者，例如开发人员、项目经理、营销人员、用户、测试人员或文档的编写人员。描述了文档中剩余部分的内容及其组织结构。提出了最适合于每一类型读者阅读文档的建议。

a.4 产品的范围

提供了对指定的软件及其目的的简短描述，包括利益和目标。把软件与企业目标或业务策略相联系。可以参考项目视图和范围文档而不是将其内容复制到这里。

a.5 参考文献

列举了编写软件需求规格说明时所参考的资料或其它资源。这可能包括用户界面风格指导、合同、标准、系统需求规格说明、使用实例文档，或相关产品的软件需求规格说明。在这里应该给出详细的信息，包括标题名称、作者、版本号、日期、出版单位或资料来源，以方便读者查阅这些文献。

b. 综合描述

这一部分概述了正在定义的产品以及它所运行的环境、使用产品的用户和已知的限制、假设和依赖。

b.1 产品的前景

描述了软件需求规格说明中所定义的产品的背景和起源。说明了该产品是否是产品系列中的下一成员，是否是成熟产品所改进的下一代产品、是否是现有应用程序的替代品，或者是否是一个新型的、自含型产品。如果软件需求规格说明定义了大系统的一个组成部分，那么就要说明这部分软件是怎样与整个系统相关联的，并且要定义出两者之间的接口。

b.2 产品的功能

概述了产品所具有的主要功能。其详细内容将在 d中描述，所以在此只需要概略地总结，例如用列表的方法给出。很好地组织产品的功能，使每个读者都易于理解。用图形表示主要的需求分组以及它们之间的联系，例如数据流程图的顶层图或类图，都是有用的。

b.3 用户类和特征

确定你觉得可能使用该产品的不同用户类并描述它们相关的特征（见第 7章）。有一些需求可能只与特定的用户类相关。将该产品的重要用户类与那些不太重要的用户类区分开。

b.4 运行环境

描述了软件的运行环境，包括硬件平台、操作系统和版本，还有其它的软件组件或与其共存的应用程序。

b.5 设计和实现上的限制

确定影响开发人员自由选择的问题，并说明这些问题为什么成为一种限制。可能的限制包括如下内容：

- 必须使用或者避免的特定技术、工具、编程语言和数据库。
- 所要求的开发规范或标准（例如，如果由客户的公司负责软件维护，就必须定义转包者所使用的设计符号表示和编码标准。
- 企业策略、政府法规或工业标准。
- 硬件限制，例如定时需求或存储器限制。
- 数据转换格式标准。

b.6 假设和依赖

列举出在对软件需求规格说明中影响需求陈述的假设因素（与已知因素相对立）。这可能包括你打算要用的商业组件或有关开发或运行环境的问题。你可能认为产品将符合一个特殊的用户界面设计约定，但是另一个 SRS读者却可能不这样认为。如果这些假设不正确、不一致或被更改，就会使项目受到影响。

此外，确定项目对外部因素存在的依赖。例如，如果你打算把其它项目开发的组件集成到系统中，那么你就要依赖那个项目按时提供正确的操作组件。如果这些依赖已经记录到其它文档(例如项目计划)中了，那么在此就可以参考其它文档。

c. 外部接口需求

利用本节来确定可以保证新产品与外部组件正确连接的需求。关联图表示了高层抽象的外部接口。需要把对接口数据和控制组件的详细描述写入数据字典中。如果产品的不同部分有不同的外部接口，那么应把这些外部接口的详细需求并入到这一部分的实例中。

c.1 用户界面

陈述所需要的用户界面的软件组件。描述每个用户界面的逻辑特征。以下是可能要包括的一些特征：

- 将要采用的图形用户界面（GUI）标准或产品系列的风格。
- 屏幕布局或解决方案的限制。
- 将出现在每个屏幕的标准按钮、功能或导航链接（例如一个帮助按钮）。
- 快捷键。
- 错误信息显示标准。

对于用户界面的细节，例如特定对话框的布局，应该写入一个独立的用户界面规格说明中，而不能写入软件需求规格说明中。

c.2 硬件接口

描述系统中软件和硬件每一接口的特征。这种描述可能包括支持的硬件类型、软硬件之

间交流的数据和控制信息的性质以及所使用的通信协议。

c.3 软件接口

描述该产品与其它外部组件（由名字和版本识别）的连接，包括数据库、操作系统、工具、库和集成的商业组件。明确并描述在软件组件之间交换数据或消息的目的。描述所需要的服务以及内部组件通信的性质。确定将在组件之间共享的数据。如果必须用一种特殊的方法来实现数据共享机制，例如在多任务操作系统中的一个全局数据区，那么就必须把它定义为一种实现上的限制。

c.4 通信接口

描述与产品所使用的通信功能相关的需求，包括电子邮件、 Web浏览器、网络通信标准或协议及电子表格等等。定义了相关的消息格式。规定通信安全或加密问题、数据传输速率和同步通信机制。

d. 系统特性

在图9-1所示的模板中，功能需求是根据系统特性即产品所提供的主要服务来组织的。你可能更喜欢通过使用实例、运行模式、用户类、对象类或功能等级来组织这部分内容（ IEEE 1998）。你还可以使用这些元素的组合。总而言之，你必须选择一种使读者易于理解预期产品的组织方案。

仅用简短的语句说明特性的名称，例如“ 4.1拼写检查和拼写字典管理 ”。无论你想说明何种特性，阐述每种特性时都将重述从d.1～d.3这三步系统特性。

d.1 说明和优先级

提出了对该系统特性的简短说明并指出该特性的优先级是高、中，还是低。或者你还可以包括对特定优先级部分的评价，例如利益、损失、费用和风险，其相对优先等级可以从 1（低）到9（高）（见第13章）。

d.2 激励/响应序列

列出输入激励（用户动作、来自外部设备的信号或其它触发器）和定义这一特性行为的系统响应序列。就像在第8章讨论的那样，这些序列将与使用实例相关的对话元素相对应。

d.3 功能需求

详列出与该特性相关的详细功能需求。这些是必须提交给用户的软件功能，使用户可以使用所提供的特性执行服务或者使用所指定的使用实例执行任务。描述产品如何响应可预知的出错条件或者非法输入或动作。就像本章开头所描述的那样，你必须唯一地标识每个需求。

e. 其它非功能需求

这部分列举出了所有非功能需求，而不是外部接口需求和限制。

e.1 性能需求

阐述了不同的应用领域对产品性能的需求，并解释它们的原理以帮助开发人员作出合理的设计选择。确定相互合作的用户数或者所支持的操作、响应时间以及与实时系统的时间关系。你还可以在这里定义容量需求，例如存储器和磁盘空间的需求或者存储在数据库中表的最大行数。尽可能详细地确定性能需求。可能需要针对每个功能需求或特性分别陈述其性能需求，而不是把它们都集中在一起陈述。例如，“ 在运行微软Window 2000的450 MHzPentium Ⅱ的计算机上，当系统至少有50%的空闲资源时，95%的目录数据库查询必须在两秒内完成 ”。

e.2 安全设施需求

详尽陈述与产品使用过程中可能发生的损失、破坏或危害相关的需求。定义必须采取的安全保护或动作，还有那些预防的潜在的危险动作。明确产品必须遵从的安全标准、策略或规则。一个安全设施需求的范例如下：“如果油箱的压力超过了规定的最大压力的 95%，那么必须在1秒钟内终止操作”。

e.3 安全性需求

详尽陈述与系统安全性、完整性或与私人问题相关的需求，这些问题将会影响到产品的使用和产品所创建或使用的数据的保护。定义用户身份确认或授权需求。明确产品必须满足的安全性或保密性策略。你可能更喜欢通过称为完整性的质量属性来阐述这些需求，完整性将在第11章介绍。一个软件系统的安全需求的范例如下：“每个用户在第一次登录后，必须更改他的最初登录密码。最初的登录密码不能重用。”

e.4 软件质量属性

详尽陈述与客户或开发人员至关重要的其它产品质量特性（见第 11章）。这些特性必须是确定、定量的并在可能时是可验证的。至少应指明不同属性的相对侧重点，例如易用程度优于易学程度，或者可移植性优于有效性。

e.5 业务规则

列举出有关产品的所有操作规则，例如什么人在特定环境下可以进行何种操作。这些本身不是功能需求，但它们可以暗示某些功能需求执行这些规则。一个业务规则的范例如下：“只有持有管理员密码的用户才能执行 $100.00或更大额的退款操作。”

e.6 用户文档

列举出将与软件一同发行的用户文档部分，例如，用户手册、在线帮助和教程。明确所有已知的用户文档的交付格式或标准。

f. 其它需求

定义在软件需求规格说明的其它部分未出现的需求，例如国际化需求或法律上的需求。你还可以增加有关操作、管理和维护部分来完善产品安装、配置、启动和关闭、修复和容错，以及登录和监控操作等方面的需求。在模板中加入与你的项目相关的新部分。如果你不需要增加其它需求，就省略这一部分。

附录A：词汇表

定义所有必要的术语，以便读者可以正确地解释软件需求规格说明，包括词头和缩写。你可能希望为整个公司创建一张跨越多项项目的词汇表，并且只包括特定于单一项目的软件需求规格说明中的术语。

附录B：分析模型

这个可选部分包括或涉及到相关的分析模型的位置，例如数据流程图、类图、状态转换图或实体-关系图（见第10章）。

附录C：待确定问题的列表

编辑一张在软件需求规格说明中待确定问题的列表，其中每一表项都是编上号的，以便于跟踪调查。

9.3 编写需求文档的原则

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（Kovitz 1999）。你在编写软件需求文档时，应牢记以下几点建议：

- 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化学药品容器清单。”
- 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、容易、简单、迅速、有效、支持、许多、最新技术、优越的、可接受的和健壮的。当用客说“用户友好”或者“快”或者“健壮”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，例如：提高、最大化、最小化和最佳化。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。含糊的语句表达将引起需求的不可验证。

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。编写详细的需求文档，所带来的益处是如果需求得到满足，那么客户的目的也就达到了，但是不要让过于详细的需求影响了设计。如果你能用不同的方法来满足需求且这种方法都是可接受的，那么需求的详细程度也就足够了。然而，如果评审软件需求规格说明的设计人员对客户的意图还不甚了解，那么就需要增加额外的说明，以减少由于误解而产生返工的风险。

需求文档的编写人员总是力求寻找到恰如其分的需求详细程度。一个有益的原则就是编写单个的可测试需求文档。如果你想出一些相关的测试用例可以验证这个需求能够正确地实现，那么就达到了合理的详细程度。如果你预想的测试很多并且很分散，那么可能就要将一些集合在一起的需求分离开。已经建议将可测试的需求作为衡量软件产品规模大小的尺度（Wilson 1995）。

文档的编写人员必须以相同的详细程度编写每个需求文档。我曾见过在同一份软件需求规格说明中，对需求的说明五花八门。例如，“组合键Control-S代表保存文件”和“组合键Control-P代表打印文件”被当成两个独立的需求。然而，“产品必须响应以语音方式输入的编辑指令”则被作为一个子系统，而并不作为一个简单的功能需求。

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。

文档的编写人员在编写软件需求规格说明时不应该出现需求冗余。虽然在不同的地方出现相同的需求可能会使文档更易读，但这也造成了维护上的困难。需求的多个实例都需要同时更新，以免造成需求各实例之间的不一致。在软件需求规格说明中交叉引用相关的各项，

在进行更改时有助于保持它们之间的同步。让独立性强的需求在需求管理工具或数据库中只出现一次，这样可以缓和冗余问题。

表9-1 适合某种模式的需求编号列表的表格化示例

管 区	带标记格式	无标记格式	ASCII格式
联邦	ED-13.1	ED-13.2	ED-13.3
州	ED-13.4	ED-13.5	ED-13.6
地区	ED-13.7	ED-13.8	ED-13.9
国际	ED-13.10	ED-13.11	ED-13.12

文档的编写人员应考虑用最有效的方法表达每个需求。考虑符合如下句型的一系列需求：“文本编辑器应该能分析定义有<管区>法律的<格式>文档。”对于12种相似的需求中，<格式>所取的可能值有3种，<管区>所取的可能值有4种。当你评审与此类似的需求时，很难发现遗漏了一个需求，例如“文本编辑器应该能分析定义了国际法的无标记文档。”就像表9-1所示那样，可以用表中的这种格式表示需求，以确保你没有遗漏掉任何一个需求。更高层的需求可以这样描述：“ED-13文本编辑器应该能分析定义有不同管区法律的多种格式的文档，如表<xx>所示。”

9.4 需求示例的改进前后

第1章曾经提出了高质量需求陈述的许多特性：完整性、正确性、可行性、必要性、设定优先级、明确性和可验证性。由于不具有这些特征的需求将会引起混淆，导致将来的返工，因此，你必须尽快找出并纠正这些问题。下面是从真实项目中改编一些存在问题的需求。根据这些质量特性来检查每一个需求陈述，看一看你能否发现问题所在。针对每个需求陈述，将给出我的看法和建议性的改进方案。我确信，体验每个改进方案将使你受益匪浅，但你的目标不是编写完美的需求文档。你只需在一个可接受的风险程度上编写需求文档，使之有助于你的设计和软件构造。

例1

“产品必须在固定的时间间隔内提供状态消息，并且每次时间间隔不得小于 60秒”。

这个需求看起来是不完整的：什么是状态消息，并且在什么情况下向用户提供这些消息？显示时间多长？我们所说的是产品的哪一部分？时间间隔也会导致混淆。假定显示状态消息之间的时间间隔只要求不少于 60秒，按这样推理，是否可以每隔一年显示一次状态信息？如果意图是消息之间的时间间隔不多于 60秒，那么1毫秒会不会太短？消息显示的时间间隔怎样才能一致？“每次”这个词混淆了这一问题。由于这些问题的存在，导致了需求是不可验证的。

这里提出一个方法使我们能重写需求文档来表述这些缺点（从我们与客户一致的目标出发作出一些猜测）即：后台任务管理器（BTM）应该在用户界面的指定区域显示状态消息。

a. 在后台任务进程启动之后，消息必须每隔 60（±10）秒更新一次，并且保持连续的可见性。

b. 如果正在正常处理后台任务进程，那么后台任务管理器（ BTM）必须显示后台任务进程已完成的百分比。

c. 当完成后台任务时，后台任务管理器（BTM）必须显示一个“已完成”的消息。

d. 如果后台任务中止执行，那么后台任务管理器（BTM）必须显示一个出错消息。

我把原先的需求分割成多个需求，因为每个需求都需要独立的测试用例并且使各个需求都具有可跟踪性。如果把多个需求都集中在一个段落中，那么在构造软件和测试时就很容易忽略其中某个需求。注意，修改之后的需求并没有精确地说明是怎样显示状态信息的。这是一个设计问题，如果你在这个地方详述该问题，那么就会给开发人员带来设计上的一些限制。过早地限制设计上的可选方案将会给编程人员带来不利因素，并可导致产品设计的失败。

例2

“产品必须在显示和隐藏非打印字符之间进行瞬间切换”。

在瞬间这一时间概念上，计算机不能完成任何工作，因此，这个需求是不可行的。该需求也是不完整的，因为它没有说清状态切换的原因。在特定的条件下，软件产品是否可以进行自动切换或者可否由用户采取某些措施来激发这样转变？还有，在文档中显示转变的范围是什么？是所选的文本、整个文档或其它内容？这个需求也存在一个不确定性问题。“非打印”字符是否指隐藏文本、属性标记或者其它的控制字符？由于存在这些问题，该需求是不可验证的。

用如下的语句描述这个需求可能会更好一些：“用户在编辑文档时，通过激活特定的触发机制，可以在显示和隐藏所有HTML标记之间进行切换”。现在，指代关系就清楚了，非打印字符指的是HTML标记。修改过的需求指明了是用户触发了显示状态的转换，但是它并没有对设计造成限制，因为它并没有精确定义所使用的机制。当设计人员选择好一种触发机制（例如热键、菜单命令或语音输入）时，你就可以编写详细的测试用例来验证这种转换操作是否正确。

例3

“分析程序应该能生成HTML标记出错的报告，这样就可以使HTML的初学者使用它来迅速排错。”

“迅速”这个词具有模糊性。缺乏对出错报告内容的定义表明该需求是不完整的。我不知道你是如何验证这个需求的。找一些HTML的初学者，看他们利用这个报告是否可以迅速排错？还有一点不清楚的是：HTML初学者使用的是分析程序还是出错报告。并且何时生成这样的报告？

让我们使用另一种方式表述这个需求：

a. 在HTML分析程序完全分析完一个文件后，该分析程序必须生成一个出错报告，这个报告中包含了在分析文件过程中所发现错误的HTML所在的行号以及文本内容，还包含了对每个错误的描述。

b. 如果在分析过程中未发现任何错误，就不必生成出错报告。

现在我们知道了任何生成出错报告及其所包含的内容，但是我们已经把该需求提交给设计人员，让他们来决定报告的形式。我们还指明了一种例外情况：如果没有任何错误，就不生成出错报告。

例4

“如果可能的话，应当根据主货物编号列表在线确认所输入的货物编号。”

我在想：“如果可能的话”这句话意味着什么？该需求是否在技术上可行？是否可以在线访问主货物编号列表？如果你不能确信是否可以递交一个请求，那么就使用“待确定”(TBD)

来表示未解决的问题。这个需求是不完整的，因为它并没有指明如果确认通过或失败，将会发生什么情况。应该尽量避免使用不精确的词汇，例如“应当”。客户可能需要这个功能或者不需要这个功能。一些需求规格说明利用关键字之间微妙的差别如“应当”，“必须”和“可能”来指明重要性。我更喜欢使用“必须”或“将要”来明确说明需求的目的并且明确指定其优先级。

改进后的该需求描述如下：

“系统必须根据在线的主货物编号列表确认所输入的货物编号。如果在主列表中查不到该货物的编号，系统必须显示一个出错消息并且拒绝订货。”

第二种相关需求可能记录了一种异常情况：当进行货物编号确认时，主货物编号列表不可访问。

例5

“产品不应该提供将带来灾难性后果的查询和替换选择。”

“灾难性后果”的含义是解释的中心词。在编辑文档时，毫无目的地作出全局性变化而用户又不能检测出错误或没有任何办法来纠正它，此时就可能带来灾难性后果。你也要合理地使用反面需求，因为这些需求描述了系统所不能做的事情。潜在的关注焦点在于当发生意外损坏时，能保护文件的内容。也许，真正的需求是针对多级撤销 (undo)能力、全局变化或其它可导致数据丢失行为确定的。

9.5 数据字典

在很久以前，我曾经参与一项项目，在此期间，由三位编程人员对于同一数据项使用不同的变量名称、长度和有效性验证。这将导致在真正的数据定义上的混淆 ,当数据存入占据空间太小的变量中时将要截短数据，并且造成维护上的困难。我们所缺少的是一个数据字典——一个定义应用程序中使用的所有数据元素和结构的含义、类型、数据大小、格式、度量单位、精度以及允许取值范围的共享仓库。

数据字典可以把不同的需求文档和分析模型紧密结合在一起。如果所有的开发人员在数据字典上取得一致意见，那么就可以缓和集成性问题。为了避免冗余和不一致性，应该为你的项目创建一个独立的数据字典，而并不是在每个需求出现的地方定义每一个数据项。数据字典的维护独立于软件需求规格说明，并且在产品的开发和维护的任何阶段，各个风险承担者都可以访问数据字典。

在数据字典中，可以使用简单的符号表示数据项（Robertson and Robertson 1994）。项写在等号的左边，而其定义写在右边。这种符号定义了原数据元素、组成结构体的复杂数据元素、重复的数据项、一个数据项的枚举值以及可选的数据项。以下所示的例子来自“化学制品跟踪系统”。

1) 原数据元素　一个原数据元素是不可分解的。可以给它赋予一个数量值。原数据的定义必须确定其数据类型、大小、允许取值的范围等等。典型的原数据元素的定义是一行注释文本，并以星号作为界限：

请求标识号 =*6位系统生成的顺序整数，以1开头，并能唯一标识每个请求*

2) 组合项　一个数据结构或记录包含了多个数据项。如果数据结构中的项是可选的，就把它用括弧括起来：

请求的化学制品=化学制品标识号
+数量
+数量单位
+（供应商名称）

这个结构确定了与请求一种特定化学制品相关的所有信息。供应商名称是可选的，因为提出请求的人并不关心化学制品是从哪个供应商处购买来的。每个出现在结构中的数据项都必须写入数据字典。结构中还可以包含其它结构。

3) 重复项　如果一个项的多个实例将出现在数据结构中，就把该项用花括弧括起来。如果你知道可能允许的重复次数，就用“最小值：最大值”这种形式写在括号之前：

请求=请求标识号
+产品编号
+1:10{请求的化学制品}

这个例子表明，一个化学制品的请求至少应包含一种化学制品，但不能多于 10种。每个请求也包括一个单一的请求标识号和一个产品编号，它们的格式将在数据字典的其它地方定义。

4) 选择项　如果一个原数据项元素可以取得有限的离散值，就把这些值列举出来：

数量单位＝［“克”|“千克”|“个”］
* 文本串表示了与所请求的化学制品的量相关的单位 *

表明了数量单位的文本串只允许 3 种取值。注释提供了数据项定义的信息。

你在创建数据字典和词汇表上所花费的时间可以大大减少由于项目的参与者对一些关键信息的理解不一致所带来时间的浪费。如果你保持词汇表和数据字典的正确性，那么在系统的整个维护期间和相关的及以后产品的开发中，它们将是很有价值的工具。

下一步：

- 从你的软件需求规格说明中取出一页功能需求说明。检查每个语句，看它是否与好的需求特性相符，重写那些不符合的需求。
- 如果公司对需求文档没有标准的格式，那么就召集一个小的工作组讨论决定采纳的一个标准的软件需求规格说明模板。从图 9-1的模板开始并改编这个模板，使其最好地适合你的项目和产品。在标识需求方面也要一致。
- 召集一个由3到7个项目的风险承担者组成的小组来正式评审项目中的软件需求规格说明。确保每个需求规格说明都是清晰的、可行的、可验证的及无二义性的等等。寻找规格说明中不同需求间的冲突，以及软件需求规格说明中遗漏的部分。通过检查软件需求规格说明，确保纠正了在软件需求规格说明中及其后续产品里出现的错误。