# 第15章　需求开发向设计规划的转化

虽然在开发的开始阶段困难重重，但是“化学制品跟踪系统”的开发工作总算进展顺利。项目的主办者 Gerhard和化学制品仓库的产品代表者 Roxanne怀疑是否有必要花费这么多时间来收集需求。然而，他们很愿意同开发小组和其它产品代表一起参加为期一天的软件需求方面的培训。这次培训着重讲述在编写软件程序之前，使项目所有风险承担者在对需求的理解上达成共识的重要性。在培训课中，所有的成员了解了需求的术语、概念和他们要用到的方法，这可以激发他们在具体需求实施过程中采用一些改进需求的技术。

在项目开发过程中，Gerhard收到了用户代表关于如何进行需求开发的一些良好反馈。后来，他请开发组成员和产品代表者共进午餐，以庆祝他们在“化学制品跟踪系统”确定需求基线方面达到了一个重要的里程碑。在就餐时， Gerhard满怀热情地感谢了参与获取需求的人员所做出的贡献以及他们的合作，然后他继续说，“现在你们有了完整正规的需求，我盼望开发小组很快能写出程序代码。”

“我们并不准备马上编写程序代码”，项目经理提醒Gerhard。“我们打算分阶段发行产品，所以我们需要想出一个最好的方法来设计系统，使其能适应将来系统的扩展。我们可以从产品原型中获得关于有效技术途径的一些思想，并且产品原型有助于我们理解用户喜欢的界面特性。如果现在我们在软件的设计上多花上一点时间，那么到明年我们扩充产品的更多功能时就不会遇到较大的问题。”

这时Gerhard有一点沮丧。因为他觉得开发人员现在就可以开始编制程序了。但是Gerhard是否有点行动过早？

有经验的项目经理和开发人员知道把软件需求转化为健壮的设计和合理的项目规划的重要性。本章通过需求和项目规划、设计、编码和测试之间的联系来探讨需求开发和一个成功发行的产品之间的转换方法。

## 15.1　从需求到项目规划

由于需求定义了项目预期的成果（ outcome），所以你的项目规划、预测和进度安排都必须以软件需求为基础。

### 15.1.1　需求和进度安排

许多软件工程实行“从右到左的进度安排”，此时，规定了发行产品的具体日期而后定义产品的需求。当开发者要实现预期质量标准下所有要求的功能时，他们常常不能按时完成项目。在做出详细的规划和约定之前定义软件需求是更现实的。然而，如果你在需求的哪些部分能适应进度安排的限制哪些部分不能适应进度安排的限制这一问题上还有商量余地的话，那么“从设计到进度安排”的策略是可以起作用的。

对于复杂的系统，软件仅是最终产品的一部分时，只有在产品级（系统）需求产生以后，才能建立高层的进度安排。然后，将系统需求分解并分配到各个不同的软硬件子系统中。从

这一点上看，就可以以不同来源（包括市场、销售、客户服务以及开发）的输入为基础建立起一致的产品发行日期。如果存在进度安排的约束条件，那么具有交叉功能的开发小组必须在功能、质量和费用上作出合理（trade-off）的决策。

你可能考虑按阶段规划和提供项目资金。需求探索作为第一阶段将提供足够的信息，使你能为一个或更多的构造阶段进行现实的规划和预测。具有不确定需求的项目也可以从反复或渐增的软件开发生存期中得以改善。定义需求的优先级可以使你判断出哪些功能应包括在首发版中，哪些功能放到随后发行的版本中。

软件项目可能经常不能达到预定的目标，这是因为开发人员和其他项目参与者是拙劣的规划者，倒不是因为他们是拙劣的软件工程师。主要的规划失误包括：忽略公共（用）的项目任务，低估了要花费的工作量和时间，没有考虑项目风险，并且没有考虑返工所需的时间。正确的项目规划需要以下元素：

- 根据对需求的清楚理解来估计产品规模的大小。
- 根据历史记录了解开发小组的工作效率。
- 需要一张综合的任务列表以完整实现和验证每一特性或使用实例。
- 有效的预测和规划过程。
- 经验。

### 15.1.2 需求和预估

项目估计（预估）的第一步就是要把需求和软件产品规模的大小相联系。你可以根据文本需求、图形分析模型、原型或用户界面设计来预估产品的大小。虽然对于软件大小没有完善的度量标准，但以下给出了一些常用的度量标准：

- 功能点和特性点的多少（Jones 1996b），或者3-D功能点的数量(Whitmire 1995)。
- 图形用户界面（GUI）元素的数量、类型和复杂度。
- 用于实现特定需求所需的源代码行数。
- 对象类的数量或者其它面向对象系统的衡量标准（ Whitmire 1997）。
- 单个可测试需求的数量（Wilson 1995）。

所有这些方法都可用于预估软件的大小，但不管采用什么方法你都必须根据经验进行选择。如果你没有记录当前项目所完成的真正结果，并和你所预估的进行比较，利用那些知识来提高你的预估能力，那么你所预测的将永远是一种猜测。积累数据是需要时间的，所以你可以把度量软件大小的标准与实际的开发工作量相联系。你的目标是建立可以使你从需求文档中预估整个软件大小的方程（等式）或者借助归纳等解决问题的方法。另一种方法是从需求中预测代码行、功能点或图形用户界面元素的数量，并使用商业预估软件作出人员水平与开发进度的合理安排。你可以从 http://www.method-tools.com获得一些可用的预估工具的信息。

含糊的和不确定的需求必定会引起你在软件大小预估中的不确定性，从而导致你的工作量和进度安排预估的不确定。因为在项目的早期阶段，需求的不确定性是不可避免的，所以在进度安排中应包括临时的事件并要合理预算资金以适应一些需求的增加和可能的超限。一个合理的住房改造项目包括了对不可预料的偶然事件，在你的项目中是否也要如法炮制呢？

花一点时间考虑一下，哪些标准最适合你所主持的项目。应该意识到开发时间与产品大

小或开发小组的大小没有线性关系。在产品大小、工作量、开发时间、生产率和人员技术积累时间之间是存在着复杂的关系（Putnam and Myers 1997）。理解这种关系可以防止你陷入进度安排或人员任务安排承诺的误区（以前没有类似项目成功完成过），否则，项目开发将不会成功完成。

## 15.2 从需求到设计和编码

需求和设计之间存在差别，但尽量使你的规格说明的具体实现无倾向性。理想情况是：在设计上的考虑不应该歪曲对预期系统的描述（Jackson 1995）。需求开发和规格说明应该强调对预期系统外部行为的理解和描述。让设计者和开发者参与需求审查以判断需求是否可以作为设计的基础。

不同的软件设计方法常常都会满足最终需求，而设计方法会随着性能、有效性、健壮性以及所采用的技术上的不同而变化。如果你直接从需求规格说明跳到编码阶段，你所设计的软件将会是空中阁楼，其可能的结果只能是结构性很差的一个软件。在构造软件之前，你应该仔细考虑构造系统的最有效的方法。考虑一下其它的设计方案将有助于确保开发人员遵从所提出的设计约束或遵从与设计有关的质量属性规格说明。

我曾经参与一项项目，进行了完整的需求分析，建立了详细描述模拟摄像系统行为的 8个变换过程的数据流程图。经过大量的需求分析后，我们并没有直接进行源代码的编写工作。而是以数据流程图为表示方法，创建了一个设计模型。我们立刻意识到模型中有三个步骤使用了相同的计算算法，另外三个使用不同的方程集，而剩下的两个步骤共享三分之一集合。

分析模型代表了用户和开发小组对我们正在解决的问题的理解，而设计模型则描绘了我们应该如何构造系统。通过设计期间的仔细思考，我们把核心问题简化了 60%，把8个复杂的计算集合减少到3个。如果我们在需求分析之后立刻进行编码，那么在构造阶段必定会出现代码重复。但是，由于及早发现了可简化问题，我们节省了许多时间和金钱。设计上的返工比编码返工可能要效率高一些。

以需求为基础，反复设计将产生优良成果。当你得到更多的信息或额外的思想时，用不同的方法进行设计可以精细化你最初的概念。设计上的失误将导致软件系统难以维护和扩充，最终会导致不能满足客户在性能和可靠性上的目标。在把需求转化为设计时你所花的时间将是对建立高质量、健壮性产品的关键的投资。

在设计产品时，产品的需求和质量属性决定了所采用的合适的构造方法（Bass, Clements and Kazman 1998）。研究和评审所提出的体系结构是另一种解释需求的方法且会使需求更加明确。与原型法类似，这是一种自下而上的需求分析方法。两种方法都围绕着这样一种思维过程：“如果我正确理解需求，那么这种方法可以满足这种需求。既然我手中有一个最初的体系结构（或原型），它是否有助于我更好地理解需求呢？”

在你可以开始实现各个部分需求前，不必为整个产品进行完整、详细的设计。然而，在你进行编码前，必须设计好每个部分。设计规划将有益于大难度项目（有许多内部组件接口和交互作用的系统和开发人员无经验的项目）（McConnell 1998）。然而，下面介绍的步骤将有益于所有的项目：

- 应该为在维护过程中起支撑作用的子系统和软件组件建立一个坚固的体系结构。
- 明确需要创建的对象类或功能模块，定义他们的接口、功能范围以及与其它代码单元的

协作。

- 根据强内聚、松耦合和信息隐藏的良好设计原则定义每个代码单元的预期功能。
- 确保你的设计满足了所有的功能需求并且不包括任何不必要的功能。

当开发者把需求转化为设计和代码时，他们将会遇到不确定和混淆的地方。理想情况下，开发者可沿着发生的问题回溯至客户并获得解决方案。如果不能马上解决问题，那么开发者所做出的任何假设，猜想或解释都要编写成文档记录下来，并由客户代表评审。如果遇到许多诸如此类的问题，那么就说明开发者在实现需求之前，这些需求还不十分清晰或具体。在这种情况下，最好安排一两个开发人员对剩余的需求进行评审后才能使开发工作继续进行。

## 15.3 从需求到测试

详尽的需求是系统测试的基础，反过来只能通过测试来判断软件是否满足了需求。你必须针对软件需求规格说明中所记录的产品的预期行为来测试整个软件，而不是针对设计或编码。基于代码的系统测试可以变成“自满足的预见（self-fulfilling prophecy）”。产品可以正确呈现基于代码的测试用例所描述的所有行为，但这并不意味着产品正确地实现了用户的需求。如果你没有文档形式的需求，你应该重新获得需求以开发合适的测试用例，这将是一个低效的和不准确的方法。让测试人员参与需求审查以确保需求是明确的，通过验证的需求才可以作为系统测试的基础。

在需求开发中，当每个需求都稳定之后，项目的系统测试人员应该编写文档，以记录他们如何验证需求 —— 通过测试、审查，演示或分析。对如何验证每一需求的思考过程本身就是一种很有用的质量审查实践。根据需求中的逻辑描述，利用诸如因果图等分析技术来获得测试用例，这将会揭示需求的二义性、遗漏或隐含的其它条件和其它问题。在你的系统测试方案中，每个需求应至少由一个测试用例来测试，这样就会验证所有的系统行为。你可以由跟踪通过测试的需求所占的比例来衡量测试进度。有经验的测试人员可以根据他们对产品的预期功能、用法、质量特性和特有行为的理解，概括出纯粹基于需求的测试。

基于规格说明的测试适用于许多测试设计策略：动作驱动、数据驱动（包括边界值分析和等价类的划分）、逻辑驱动、事件驱动和状态驱动（Poston 1996）。从正式的规格说明中很容易自动生成测试用例，但是对于更多的由自然语言描述的需求规格说明，你必须手工开发测试用例。比起结构化分析图，对象模型更易于自动生成测试用例。

在开发的进展过程中，你将通过详细的软件功能需求仔细推敲来自使用实例高层抽象的需求，并最终转化成单个代码模块的规格说明。测试方面的权威专家 Boris Beizer (1999)指出针对需求的测试必须在软件结构的每一层进行，而不只是在用户层进行。在一个应用程序中有许多代码不会被用户所访问，但这些代码却是产品基础操作所需要的。即使有些模块功能在整个软件产品中对用户都不可见，但是每个模块功能必须满足其自身的需求或规格说明要求。因此，针对用户需求来测试系统是系统测试的必要但非充分条件。

## 15.4 从需求到成功

我最近遇到一个项目，在该项目中，一个新的开发小组将针对早先的开发小组所开发的需求实现一个应用程序。这个新的开发小组一看到由 3英寸活页纸订成的软件需求规格说明就感到十分恐惧，于是立刻编写代码。在构造软件的过程中，他们没有参考软件需求规格说明，

而是根据他们对预期系统的不完整且不正确的理解，按他们自己的想像进行编码设计。因此，该项目遇到许多问题便不足为奇了。试图理解这个庞大的需求规格说明肯定会令人恐惧的，况且需求规格说明有些部分可能还不完善，但忽略了需求规格说明必定会导致失败。

我知道一个十分成功的项目，在开发项目的过程中，开发人员列出了与特定代码版本相对应的需求。项目的质量保证组通过对照需求来执行测试用例从而评价其对应的代码版本。根据测试标准，如果不满足需求的话就算是一个错误。如果不满足的需求数量超过预先给定的一个数字，或者特定的有重大影响的需求没有被满足，那么这个代码块就被拒绝了。这个项目的成功之处在于把需求文档作为何时发行产品的基础。

一个软件开发项目最终可发行的是满足客户需求和期望的软件系统。需求是从产品概念通向用户满意之路的最本质的一步。如果你不以高质量的需求作为项目规划、软件设计和系统测试的基础，那么在试图开发优秀产品的过程中将浪费大量的人力和物力。

然而，也不要成为需求过程的奴隶。避免陷入畸形分析的陷阱，此时，开发小组将花费大量的时间创建不必要的文档，并举行各种形式上的会议和评审，而并不编写任何软件代码，这将会导致项目被取消。努力在精确的规格说明与可将产品失败的风险降至可接受程度的编码之间做出明智的选择。

下一步：

- 检查你的软件需求规格说明中所有需求是否与软件设计的各元素相对应。这些可能在数据流模型、实体联系图中的表、对象类或方法，或者其它设计元素中处理。如果开发人员在编写代码前，没有进行软件设计，那么他们可能要进行软件设计方面的培训。
- 记录实现每一个产品特性或使用实例的代码行、功能点、对象类或图形用户界面（GUI）元素的数量。并且还要记录对完全实现并验证每个特性或使用实例所估计的工作量和实际的工作量。尽力获取产品大小与所花费工作量的关系，这将有助于你对将来的需求规格说明作出更精确的预估。