第 3 章　需求分析

需求问题是造成软件工程项目失败的主要原因，能否开发出高质量的软件，很大程度上取决于对要解决的问题的认识以及如何准确地表达出用户的需求。通过需求分析使得分析者深刻地理解和认识系统，并将其完全、准确地表达，其结果不仅起到沟通（用户和开发者）作用，还是后续工作的依据。本章介绍需求分析的一些基本概念，分别对需求获取技术、需求规格说明书、如何进行需求分析以及需求分析方法进行讨论，重点讨论结构化的需求分析方法。

3.1　需求分析的概念和任务

3.1.1　需求的概念

什么是需求？到目前为止还没有公认的定义。对用户来讲需求是对软件产品的解释，是用户对目标软件系统在功能、行为、性能、设计和约束等方面的期望；而开发人员所讲的需求对用户来说又像是详细设计。比较权威的定义是 IEEE 软件工程标准词汇表中的需求定义：

（1）用户解决问题或达到目标所需的条件或权能（Capability）。

（2）系统或系统部件要满足合同、标准、规范或其他正式规定文档所需具有的条件或权能。

（3）一种反映上面（1）或（2）所描述的条件或权能的文档说明。

由定义可知，需求一方面反映了系统的外部行为，另一方面反映了系统的内部特性，反映的方式是需求文档。用规范的格式表达出来的文档说明称为需求规格说明书，或者简称为“需求说明”。

3.1.2　需求的层次

需求可分解为 4 个层次：业务需求、用户需求、功能需求和非功能需求。

（1）业务需求（Business Requirement）：业务需求是反映组织机构或客户对软件高层次的目标要求。这项需求是用户高层领导机构决定的，它确定了系统的目标、规模和范围。业务需求是需求分析阶段制定需求调研计划、确定用户核心需求和软件功能需求的依据，应在进行需求分析之前确定，通常在项目定义与范围文档中予以说明。

（2）用户需求（User Requirement）：用户需求是用户使用该软件要完成的任务。要弄清这部分需求，应该充分调研具体的业务部门，详细了解最终用户的工作过程、所涉及的信息、当前系统的工作情况、与其他系统的接口等。用户需求是最重要的需求，也是最容易出现问题的部分。

（3）功能需求（Functional Requirement）：功能需求定义了软件必须实现的功能。由于用户是从完成任务的角度对软件提出需求的，通常是凌乱的、非系统化的、冗余的，开发人员无法据此编写程序。分析人员必须在充分理解用户需求的基础上，将用户需求整理成满足特定业务需求的软件功能需求。

（4）非功能需求：非功能需求是对功能需求的补充。可以分为两类：一类是用户关心的一些重要属性，如有效性、效率、灵活性、完整性、互操作性、可靠性、健壮性、可用性；另一类是对

开发者来说很重要的质量属性，如可维护性、可移植性、可复用性、可测试性。

软件需求各组成部分之间的关系如图 3-1 所示。

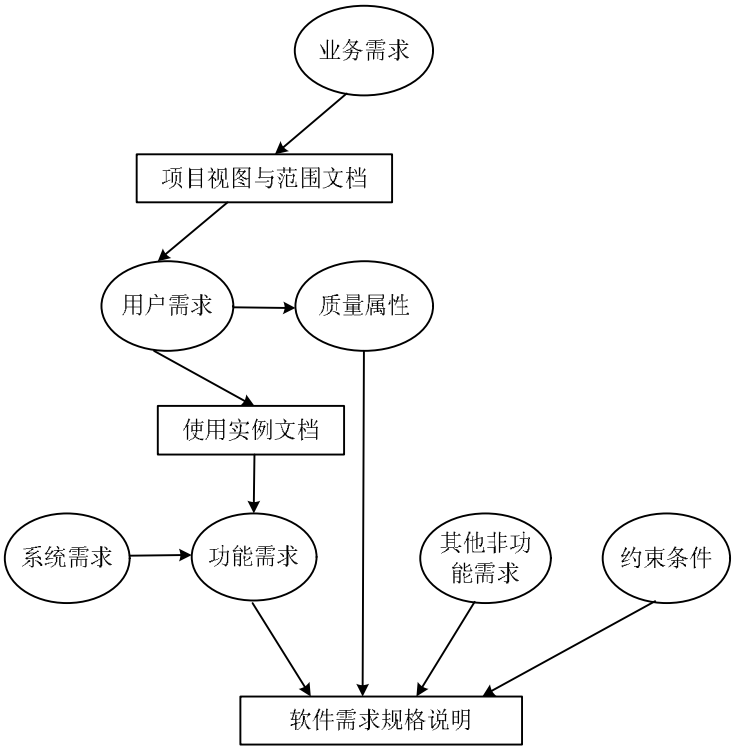


图 3-1　软件需求各组成部分之间的关系

3.1.3　需求分析的任务

软件需求分析是在软件计划的基础上进行的。需求分析的任务就是借助于当前系统的逻辑模型导出目标系统的逻辑模型，确定目标系统“做什么”的问题。需求分析的结果应该反映的是系统必须做什么，而不是怎么做。具体地讲，该阶段的工作是在对问题进行调查了解的基础上，确定系统的目标和范围，将用户的需求变为软件的功能和性能描述。为了将软件的功能和性能描述清楚，系统分析人员需要用一定的方法和手段对问题进行分析，建立反映问题所涉及的信息、功能及系统行为的模型，用文字、图形符号来详细说明软件必须要做什么以及配合运行的环境应该是什么，形成需求规格说明。

需求分析是介于系统分析和软件设计阶段之间的桥梁。一方面，需求分析以系统规格说明和项目计划作为分析活动的基本出发点，并从软件角度对它们进行检查与调整；另一方面，需求规格说明又为用户和开发人员提供一起协商讨论的基础，作为软件设计、实现、测试和维护的依据。良好的分析活动有助于避免或尽早剔除早期错误，从而提高软件生产率，降低开发成本，改进软件质量。

随着软件系统规模的扩大，需求分析与定义在整个软件开发与维护过程中越来越重要，直接关系到软件的成功与否。人们逐渐认识到需求分析活动不仅限于软件开发的最初阶段，它贯穿于系统开发的整个生命周期，并将需求工作分为需求开发和需求管理两部分，将这两部分统称为“需求工程”（Requirement Engineering，RE）。20 世纪 80 年代中期，需求工程成为软件工程的子领域，进

入 90 年代以来，需求工程成为研究的热点之一。需求工程结构图如图 3-2 所示。

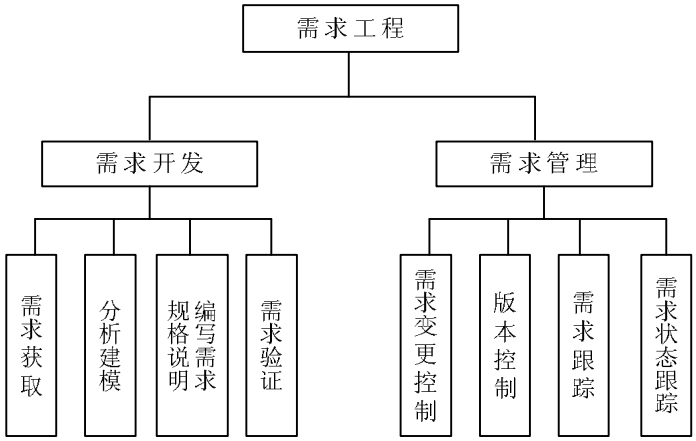


图 3-2　需求工程结构图

由此，需求分析阶段的任务就是实现需求工程，具体内容如下。

3.1.3.1　需求开发

需求开发工作包括软件产品的需求收集、评价、编写文档等所有活动，分为 4 个阶段：需求获取、分析建模、编写需求规格说明、需求验证。

1. 需求获取

进行用户需求调查，获取需求，识别问题，确定系统的综合要求是需求分析的第一步，是一个对所求解问题及其环境的理解、分析和综合的过程，也是进行软件设计与实现的基础。

在系统开发阶段的初期，分析人员往往对待解决的问题知之甚少，而用户对问题的描述、对目标软件的要求通常也相当零乱、模糊。尤其是分析人员与用户共同的知识领域不多，造成相互之间理解方面的困难。分析员通过与用户充分交流，准确、完整地获取用户需求，确定软件系统的综合要求。通常软件系统的综合要求包括以下几方面：

（1）系统功能要求：分析用户要求实现的全部功能，并分析其中每个要求的必要性和相容性。确定系统应该做什么，系统要求输入什么信息，输出什么信息，以及如何将输入变换为输出，划分出目标系统必须完成的所有功能。

（2）性能要求：理解用户对系统性能的要求，确定目标系统必须达到的技术性能指标。如响应时间，存储容量及后援存储，计算精度与效率，系统安全指标等。

（3）运行和扩充要求：合理地规定系统运行要求和系统将来可能的扩充要求。

系统运行要求通常包括系统运行的物理环境，如系统运行的设备地点，位置是集中式的还是分布式的，对环境的要求如何（如温度、湿度，电磁场干扰等）；支持软件系统；数据通信方式；系统界面，如要求与其他系统进行数据交换的内容与格式、终端用户的类型与熟练程度、用户对界面的特定要求、用户操作的易接受性等。

系统可能的扩充要求主要有是否要求可移植、未来扩充或者升级的要求、扩充的方式、范围、接口等。

（4）系统维护要求：包括系统出错后可以允许的最大恢复时间、对错误修改的回归测试要求、系统运行日志规格、是否允许对系统修改、系统变化如何反映到设计中等。

（5）系统文档规格要求：系统要求交付什么文档及各类文档的编制规范和预期使用对象等。

获取需求的方法见 3.2 节。

2. 分析建模

为了更好地识别问题，应综合上述分析的结果，对已获取的需求进行抽象描述，为目标系统建立一个详细的逻辑模型。模型是形成需求说明的重要工具，通过模型可以更清晰地记录用户对需求的表达，更方便地与用户交流，以便帮助分析人员发现用户需求中的不一致性，排除不合理的部分，挖掘潜在的用户需求，确定被开发系统的运行环境、功能和性能要求。

建模方法有多种类型，如面向过程的方法、面向数据结构的方法、面向数据流的方法及面向对象的方法等。本书后面将要介绍的分析方法，实际都是建模方法。通常系统逻辑模型可用数据流图、实体—联系图、状态转换图、数据字典和主要功能的处理算法等进行描述。

通常，软件开发是要实现系统的物理模型，即确定待开发的软件系统的系统元素，然后把功能和数据结构分配到这些系统元素中。但是，目标系统的具体物理模型是由它的逻辑模型经实例化得到的，所谓逻辑模型就是忽略实现机制和具体细节，只描述系统要完成的功能和要处理的数据。因此，软件系统的开发，可看做是建立模型及模型转换的过程。具体分析建模过程如图 3-3 所示。

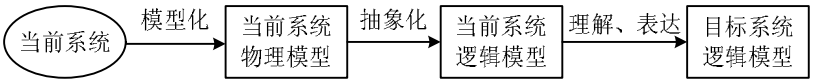


图 3-3　分析建模过程

3. 编写需求规格说明

该阶段的主要工作是需求描述。在对问题空间准确、全面理解的基础上，对需求模型进行精确地、形式化的描述。需求描述应定义和详细描述全部系统功能，反映影响系统事件前后关系的软件行为，建立系统界面的特征，并揭示设计限制。结果以文档形式表述，成为可见的，能够与用户交流的，可以进行复审的系统逻辑模型。该阶段的文档包括：需求规格说明书、用户手册初稿、确认测试计划、修改的开发计划等。其中：

（1）需求规格说明包含对目标软件系统的外部行为的完整描述、需求验证标准以及用户在性能、质量、可维护性等方面的要求。

（2）用户手册包括用户界面描述以及有关目标系统使用方法的初步构想。在需求分析阶段就开始编写用户手册，而不是在编码测试完成后组织人员另行编写，其优点在于，首先，使系统分析和设计人员在系统开发的早期就从用户的角度观察和分析系统，有利于提高系统对用户的友好程度，并有利于提高系统运行的方便性和实用性；其次，可以使用户手册随着系统各阶段的开发不断完善，有利于保证其正确性、完备性和同真实系统的一致性，并可保证其按时完成。

（3）在需求分析中确立测试标准，作为系统开发目标是否完成的验收依据。而该测试计划的依据是系统目标，包括系统功能和性能等详细内容的用户需求。由于系统目标是在需求分析阶段制定的，所以，以此为依据的确认测试计划应在同一阶段，由相同的人员制定。这样能最有效地保证两者的一致性。

（4）修改的项目开发计划是根据新的分析结果，对可行性分析和软件计划阶段中制定的初步的项目开发计划作必要的修改、补充和完善。

4. 需求验证

在以上各阶段的实施过程之中，形成了有关的文档。在此基础上，由专家、分析人员、开发人

员、用户组成评审组，对需求分析所得结果的正确性、合理性和有效性进行检查，以确保需求分析的全面性、准确性和一致性。并使用户和开发人员对需求规格说明及用户手册达成一致的理解。通常应从以下几点进行评审：

（1）完整性：完整性体现在两个方面。首先是不能遗漏任何必要的需求。避免遗漏需求的关键是需求获取的方法，后面将详细讨论这个问题。需求完整性的第二层含义是清楚、完整地描述每一项需求所要完成的任务，使开发人员理解实现这项需求的所有必要信息，用户能够审查这项需求描述的正确性。

（2）正确性：所谓正确性是指每项需求都必须准确地反映用户要完成的任务。不仅要从不同角度检查需求的正确性，还应该检查每项需求是否与软件的总体目标一致，是否超出了业务需求所定义的软件范围。

（3）一致性：用户需求必须和业务需求一致，功能需求必须和用户需求一致。严格地遵守不同层次间的一致性关系，就可保证最后开发出来的软件系统不会偏离最初的实现目标。

（4）必要性：必要性即每项需求都应该是客户所需要的，开发人员不得自作主张添加需求。检查需求必要性的方法是将每项需求回溯至用户的某项输入。

（5）无歧义性：无歧义性即需求分析结果的描述，应使不同的人员对需求的理解是一致的。如采用自然语言描述需求，其优点是任何人不经训练，都可理解，但自然语言对需求分析最大的弊病就是它的二义性。所以各种需求分析方法中都对描述语言作了一些限制，对描述语言的规定是需求分析方法的重要组成部分。发现需求二义性可通过对需求文档的正规检查，包括编写测试用例和开发原型，还可使用按多种不同的方式、从多个角度描述同一需求的方法发现需求二义性。

（6）可验证性：每项需求都应该是可验证的。系统分析员在需求分析时就要考虑每项需求的可验证性问题，为需求设计测试用例或其他的验证方法。

（7）优先级的划分：为每一项需求按照重要程度分配一个优先级，在开发产品时，可以先实现优先级最高的核心需求，将优先级低的需求放在后续版本中。优先级的划分可以帮助项目管理者解决冲突、安排阶段性交付，在必要时做出功能取舍，以最少的费用获得软件产品的最大功能。

在评审中，一旦发现问题，应尽快予以更正。若需求分析工作通过评审，则可以开始下一阶段工作。同时，需求规格说明成为用户方与系统开发方之间的合同，任何增删或改动所引起的开发规划及成本变化，应由提出方承担责任。若未通过评审，则需重新进行有关的需求分析工作。

需求开发的过程如图 3-4 所示。

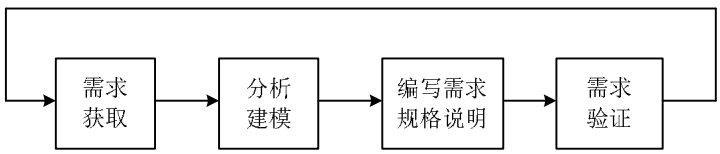


图 3-4　需求开发的过程

3.1.3.2　需求管理

由于基于计算机系统的需求常常要变更，而且变更需求的要求贯穿于整个生命期，所以，完成需求开发，形成规格说明仅是需求成功的一半，开发人员必须能够真正把所有客户的需求应用到产品中，并能够有效地控制需求变更，才能保证需求与设计的一致性，最准确地实现既定的需求。需求管理就是一组在软件开发进展中的任何时候标识、控制和跟踪需求的活动。

需求管理的内容包括在工程进展过程中为保证需求集成性及精确性所进行的所有活动，具体内

容包括需求变更控制、版本控制、需求跟踪和需求状态跟踪。

（1）需求变更控制：建议变更；评审所提出的需求变更，评估分析每项变更可能的影响，从而决定是否实施变更；以一种可控制的方式将需求变更融入到项目中。

（2）版本控制：建立需求基准版本和需求控制版本文档，确定一个需求基准。每个版本的需求规格说明都必须是独立说明，以避免将底稿和基准或新旧版本相混淆。需求文档的每一个版本必须被统一确定，在变更时，应记录变更需求文档版本的日期以及所做的变更、原因，还包括由谁负责更新和更新的新版本号等。可采用版本控制工具自动完成这些任务。

（3）需求跟踪：跟踪所有受需求变更影响的工作产品，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。当进行某项需求变更时，与其相关部分可能也需要修改。通过需求跟踪所建立的联系，可方便地找到相关的其他需求、设计模板、源代码和测试用例。通过跟踪减少当变更需求时必须进行的变更被遗漏，确保覆盖全部的需求，同时确保所有的输出符合用户的需求。

（4）需求状态跟踪：定义需求状态，在整个项目过程中跟踪需求的每一个状态及其变更情况。可将跟踪每项需求的状态建立一个数据库，其中每一条记录保存一项功能需求的重要属性。需求状态有已推荐的、已通过的、已实施的或已验证的等，这样在任何时候都能得到每个状态类的需求数量。

需求开发的结果是形成了客户与开发人员双方均满意的系统逻辑模型，它连接需求开发和需求管理，作为需求管理的输入。需求管理的过程，从需求获取即开始，并贯穿于整个软件项目生命周期，以实现最终产品同需求的最佳结合。

需求开发与需求管理间的关系可用图 3-5 来表示。

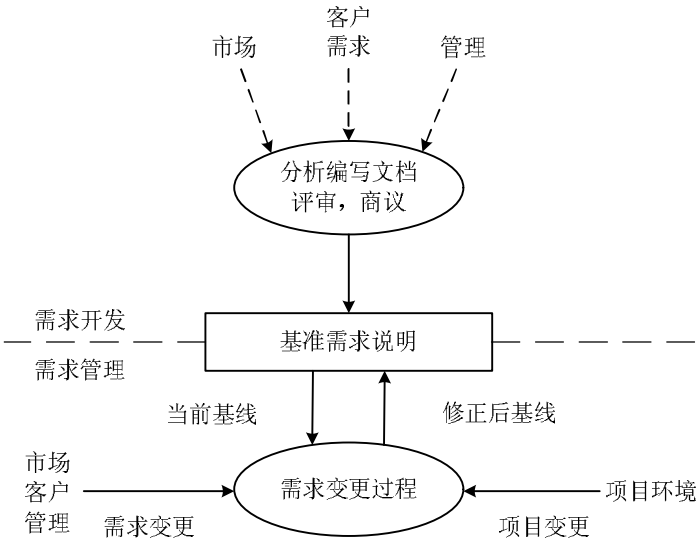


图 3-5　需求开发与需求管理间的关系

3.2　获取需求的方法

综上所述，需求分析的主要任务，就是正确理解和表达用户的要求，但如何从应用领域获取所

需知识，往往造成需求分析的障碍，所以获取需求是需求分析的关键的一步，也是最困难、最易出错的一步。

3.2.1 存在问题

在获取需求过程中遇到的典型问题有：

（1）对需求的理解问题。要准确、完整地获取需求必须对问题进行深入的理解与把握。而大多数情况下，应用领域具有一定的专业性，分析人员不是问题领域的行家，造成理解问题。

（2）分析人员与用户的通信问题。由于分析人员不懂特定的业务，需要通过用户的描述来了解。而用户对应用问题的理解、描述以及他们对目标系统的要求往往具有片面性、模糊性，在很多情况下，用户往往不能正确表达他们的需求。而且用户多是考虑业务领域本身，而分析员对问题的理解必须从信息处理要求出发，所以与用户建立相互信任，有效地沟通是分析员的首要任务。

（3）用户需求的可变性问题。应用领域与用户需求具有多样性，由于用户领域的业务不断扩展或者转移、市场竞争的要求或用户主管人员的变更等原因，使得用户的需求常发生变化。需求的可变性就要求分析员能够使其工作适应需求的变化，给需求分析造成了很大的困难。

（4）分析方法和分析工具问题。需求分析方法论和分析工具的缺乏，及其应用范围的局限性是造成障碍的又一原因。

3.2.2 常用方法

需求获取方法是沟通用户和开发人员之间的桥梁。目前，需求分析方法中，用户需求获取主要是依靠以下几种方法：

（1）访谈。

访谈是最早开始使用的获取用户需求的方法，也是目前仍然广泛使用的需求分析技术。

访谈有两种基本形式，分别是正式的和非正式的访谈。正式访谈时，系统分析员将提出一些事先准备好的具体问题，例如，询问处理的单据种类、处理的方法以及信息反馈时间应该多快等。而在非正式访谈中，分析员可提出一些用户可以自由回答的开放性问题，例如，询问用户对目前正在使用的系统有哪些不满意的地方，以鼓励被访问人员说出自己的想法。询问一个开放的、可扩充的问题将有助于更好地理解用户目前的业务过程，并且确定在新系统中应如何解决目前系统的问题。分析人员通过用户对问题的回答获取有关问题及环境的知识，逐步理解用户对目标系统的要求。

采用访谈方式时分析员的主要任务是问题的设计，包括探讨功能、非功能、例外情况的问题，甚至一些看起来“愚蠢”的问题。必须把所有的讨论记录下来，同时还要做一定的整理，并请参与讨论的用户评论并更正。

（2）问卷调查。

问卷调查即把需要调查的内容制成表格交给用户填写。该方法对需要调查大量人员的意见时，十分有效。这种方法的优点是：用户有较宽裕的考虑时间和回答时间。经过仔细考虑写出的书面回答可能比被访者对问题的口头回答更准确，从而可以得到对提出的问题较为准确细致的回答。分析员仔细阅读收回的调查表，然后再有针对性地访问一些用户，以便向他们询问在分析调查表时发现的新问题。

采用问卷调查方法的关键是调查表的设计。在开发的早期用户与开发者之间缺乏共同语言，用户可能对表格中的内容存在理解上的偏差。因此调查表的设计应简洁、易懂、易填写，同时还要注

意用户的特点和调查的策略。

（3）情景分析。

由于很多用户不了解计算机系统，对自己的业务如何在将来的目标系统中实现无认识，所以很难提出具体的需求。所谓情景分析就是对目标系统解决某个具体问题的方法和结果，给出可能的情景描述，以获知用户的具体需求。

情景分析技术的优点是，它能在某种程度上演示目标系统的行为，便于用户理解，从而进一步揭示出一些分析员目前还不知道的需求。同时，让用户起积极主动的作用对需求分析工作获得成功是至关重要的，情景分析较易为用户所理解，使得用户在需求分析过程中能够始终扮演一个积极主动的角色。因此在访问用户的过程中使用该技术是非常有效的。

（4）实地考察。

分析人员到用户工作现场，实际观察用户的手工操作过程也是一种行之有效的需求获取方法。

在实际观察过程中，分析人员必须注意，系统开发的目标不是手工操作过程的模拟，还必须考虑最好的经济效益、最快的处理速度、最合理的操作流程、最友好的用户界面等因素。因此，分析人员在接受用户关于应用问题及背景的知识的同时，应结合自己的软件开发和软件应用经验，主动地剔除不合理的、一些暂时行为的用户需求，从系统角度改进操作流程或规范，提出新的潜在的用户需求。

（5）构造原型。

在系统开发的早期，以对用户所进行的简单需求分析为基础，快速建立目标系统的原型。用户可以通过原型进行评估并提出修改意见，从而使用户明确需求。快速原型方法既可针对整个系统，也可针对系统的某部分功能。（快速原型方法内容详见 3.4 节）。

3.2.3　需求分析的原则

需求分析应遵循以下原则：

（1）解决逻辑问题。

需求分析是对问题的识别和说明过程，分析员要回答的是“系统必须做什么”的问题，而不是“系统应该怎么做”的问题。需求分析的基本原则是给出要完成的功能和处理的信息，而不考虑实现的细节，即需求分析工作集中在系统应当完成什么功能上，而不在怎样才能实现这些功能上。

（2）以运行环境为基础。

需求分析工作必须以具体的运行环境为基础，系统分析人员可以参考，但不能照搬其他类似的系统开发时的分析工作，更不能凭个人的好恶或主观想象办事。

（3）用户参与的原则。

需求分析工作是系统分析人员同用户不断交互的过程。因此，需求分析工作应该要有客户（开发委托者或开发委托者兼系统使用者）所指定的人员参加，以保证交互的充分性和工作效率。

（4）构造高质量的需求规格说明。

需求规格说明是需求分析工作重要的完成标志。在生成规格说明的有关文档的过程中，分析人员应该严格遵循既定规范，做到内容全面、结构清晰、格式严谨。

3.2.4　需求分析方法概述

随着软件开发技术的发展，目前已形成许多需求分析及描述的方法，每种方法都有其独到之处。

但不管采用哪种方法进行需求分析，都应满足如下基本要求：

（1）必须能理解问题的数据域和功能域。

一个软件从外部可以被看成是一个黑盒子，信息从一端流入，从另一端流出。在内部对输入信息进行转换，形成中间结果，再转换，最终得到所需要的输出信息，就是软件所具有的功能。所以可通过对数据的描述和数据转换的描述，实现对系统的理解和描述。

计算机程序所处理的数据域的描述一般为：数据内容、数据结构和数据流。数据内容就是数据项，数据结构就是数据项的组织形式，数据流是数据通过系统时的变化方式。

对数据进行的一系列的转换即系统应实现的功能和子功能。两个功能之间的数据传递就确定了功能之间的接口。在需求分析阶段的功能描述通常是用文字说明要“做什么”，不必具体展开怎样做。

（2）必须能按自顶向下、逐层分解的方式对问题进行分解和不断细化。

现实世界是复杂多变的，从整体上考虑一个问题通常是困难的。特别是一些复杂的问题，作为一个整体很难准确理解，不可能给出正确的解决方案。此时，对问题进行分解与抽象是普遍有效的基本法则。

分解是将求解的复杂问题，分解为若干相对简单问题求解的组合。例如实现一个教学管理系统，可以将该系统分解为学生管理、教师管理、课程管理、教务管理、考试管理等 5 个子系统。定义好各子系统之间的相互联系，对每个子系统分别求解。如子问题仍然较复杂，则可以进一步分解。如可将考试管理进一步分解为试题库维护、试卷生成、考务管理、学生考试和评阅试卷等子系统。

分解的目的是为了降低问题求解的复杂性，将复杂问题分解成一些小的、容易控制和理解的子问题不仅便于理解，还可以将子问题划分给不同的开发小组，分别完成，然后再装配起来形成一个完整的系统。更重要的是通过分解，可以促进软件开发走向构件开发的道路。因为划分的小问题中有些是常见的公共问题，有些是特殊的问题。对常见的公共问题可以复用已有的软件构件，开发人员只对特殊的问题提供解决方案。这样不仅可提高软件开发的效率，还使软件开发向着“工程化”方向迈进。

在需求阶段分解的内容，可以是软件的功能域和数据域两个部分，分解的方式有横向分解和纵向分解。横向分解在同一层次上，把一个功能分解为几个子功能，并确定这些子功能与父功能之间的接口；纵向分解是将一个功能分解为几个子功能，然后对其中复杂的子功能再继续分解为更小的子功能，逐层分解下去直到获得合适的子功能为止。在实际操作中，没有单纯的横向分解或纵向分解，通常是二者相结合的分解方式，如图 3-6 所示。

而抽象是认识问题的一般与特殊的关系。例如在上述的考试管理子系统中，可以考虑考试要求的不同试题类型，构造每种类型的典型试题，通过对典型试题的答题要求和阅卷判定方法分析，抽象出各类试题的不同答题模式和计算机阅卷策略与算法。

问题分解与抽象定义了问题的层次结构，应该在问题求解中反映出这种层次结构。结构与问题求解结构的对应关系保证了问题定义的完整性、正确性和可跟踪性。

（3）要给出系统的逻辑视图和物理视图。

软件需求的逻辑视图描述的是软件要达到的功能和要处理的信息之间的关系，但没有描述实现的细节。如库存系统中的检查库存的功能，在逻辑视图中只关心库存文件的数据结构，而不考虑计算机的具体存储方式。软件需求的逻辑描述是软件设计的基础。

软件需求的物理视图给出的是处理功能和信息结构的实际表现形式，需考虑实际的环境和具体

的设备。如一些数据是由终端键盘输入的，而有些数据可能是由模—数转换设备提供的。软件分析人员必须弄清已确定的系统元素对软件的限制，并考虑功能和信息的物理表示。注意此时功能和信息的物理表示只限于“系统必须做什么”的范围，而不考虑“如何做”的细节。

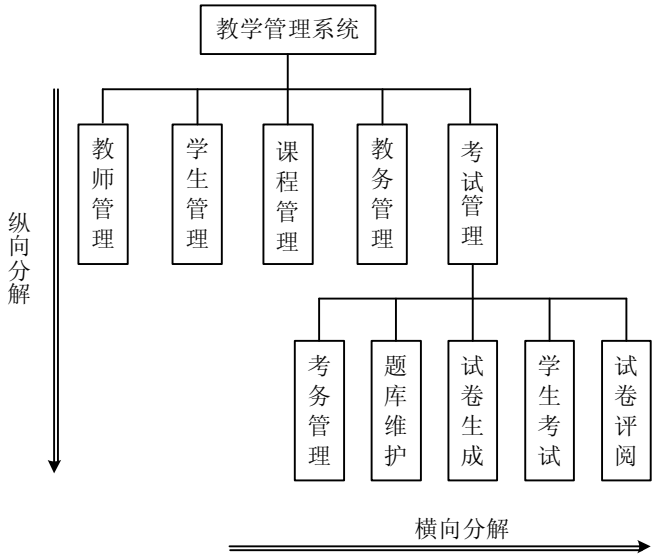


图 3-6　问题的分解

由上述讨论可见，系统的逻辑视图和物理视图描述了系统满足处理需求所提出的逻辑限制条件和系统中其他成分提出的物理限制条件。

3.3　结构化分析方法

结构化方法是 20 世纪 70 年代初，由 E.Yourdon、L.Constantine、T.DeMarco 等人提出的一种系统的软件开发方法，包括结构化分析（SA）、结构化设计（SD）和结构化编程（SP）。结构化分析方法，多年来被广泛应用，是最经典的面向数据流的需求分析方法。适用于分析大型的数据处理系统。

3.3.1　结构化分析方法的基本思想

SA 方法以数据流分析作为需求分析的出发点，任何信息处理过程均看作是将输入数据变换成所要求的输出信息的装置。而当分析人员面对一个复杂的问题时，结构化分析的策略是基于问题分解与抽象的观点，用抽象模型概念，按照软件内部的数据传递关系，采用自顶向下、逐层分解技术，直至找到满足功能需求的可实现软件元素为止。

该方法的特点是利用数据流图来帮助人们理解问题，对问题进行分析，即利用图形工具来模拟数据处理过程。该方法的核心是数据流图。数据流图是一种用来表示信息流程和信息变换过程的图解方法，它将系统看成是由数据流联系的各种功能的组合。数据流图可以方便地描述由数据流的流动联系的各种功能。通过各种功能的输入/输出结果，表现现有系统或待开发系统的功能。

具体做法是首先将整个系统看作一个加工信息处理的装置，是一个黑匣子，标识出系统边界和

所有输入输出数据流。然后自顶向下，对加工内部进行细化，逐层分解，将复杂功能分解为若干简单功能的有机组合，绘制数据流图，并逐步补充细节描述。通过将系统分解成多层处理后，在较低层次上，可以看到数据流图的高层次加工的细节和相关的数据流。用数据字典精确定义数据流图中每个数据流的成分及每个成分的属性、数据结构、数量、传递方式等。用结构化英语、判定树和判定表对数据流图中的每个加工的功能进行描述。

结构化分析方法的实质就是一种强烈依赖数据流的自顶向下的建模方法，采用一组分层的数据流图及相应的数据字典作为系统的逻辑模型。它不仅是需求分析技术，也是完成规格说明文档的技术手段。

3.3.2 描述工具

SA 方法提供一套图形、表格和结构化语言等半形式化的描述方式表达需求，简明易懂。描述工具包括：

（1）数据流图（Data Flow Diagram，DFD）：描绘系统逻辑模型的图形工具，描述了系统的组成部分及各部分之间的联系。通常通过对系统的分解得到一套分层的数据流图。

（2）数据字典（Data Dictionary，DD）：DFD 只描绘信息在系统中的流动和处理情况，而数据字典则是对图中的元素进行定义。

（3）结构化英语、判定表和判定树：详细描述数据流图中一些复杂处理的加工逻辑。

3.3.3 数据流图

SA 方法使用数据流图从数据传递和加工的角度，以图形的方式刻画数据流从输入到输出的传输变换过程。数据流图是结构化系统分析的主要工具，它表示了系统内部信息的流向，并表示了系统的逻辑处理的功能，是一种功能模型，如图 3-7 所示。

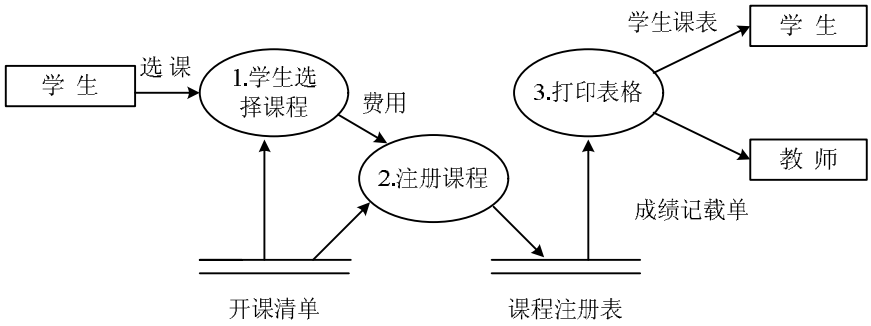


图 3-7　一个学生选课系统的 DFD

3.3.3.1 基本符号

数据流图中的基本图元包括：

○：圆框，表示加工。

□：方框，表示数据的源点或数据的终点。

→：箭头，表示被加工数据的路径和流向，即数据流。

=：双杠，表示数据存储，可以是一个记录或一个数据文件，可用名词或名词性短语命名。

1. 数据源点和数据终点

数据源点和数据终点用方框表示，它是系统之外的实体，可以是人、物、部门或者其他系统，如图 3-7 中的教师和学生。数据源点是数据流的起点，数据终点是系统数据流的最终目的地。利用数据源点与终点明确标识出系统与环境的接口，给出系统有效作用范围的边界。

2. 加工（数据处理变换）

加工用圆框表示，是对数据进行处理的逻辑单元。它接受若干输入数据流，通过加工内部产生规定的输出数据流。数据流图中对加工的标识通常由加工编号和加工命名组成。如图 3-7 中的学生选择课程、注册课程和打印表格都是加工的例子。为了给读者理解系统提供有意义的信息，对加工的命名通常要求使用“动词+宾语”形式的结构化语言简单明确地进行标识。例如“注册课程”,“打印报表”等，不应使用如“计算”、“处理”、“加工”等抽象名词作为加工命名。如果一个加工很难给出适当的命名，应该考虑该加工的处理功能是否恰当，是否应该重新分解。

3. 数据流

数据流用带数据流标识的箭头表示，表示系统处理的数据对象和数据流动的方向。数据流的方向可以是：从一加工流向另一加工；从加工流向数据存储或数据存储流向加工；从源点流向加工或从加工流向终点。

当数据流的方向指向一个加工时，表示它是该加工的一个输入数据流；当数据流的方向是从一个加工发出时，表示它是通过该加工得到的一个输出数据流。数据流是客观世界的实体对象的逻辑表示，可以是一个数据项，也可以是一组数据项组成。例如图 3-7 中“费用”由“学生学号”、“注册课程号”和“金额”组成。一个数据项可以是基本项（不可再分解的数据项，如组成“费用”的数据项），也可以是结构型数据项。例如“选课单”由“学生学号”和“注册课程列表”组成，“注册课程列表”由若干“注册项”构成，每个“注册项”由“课程名”、“课程号”、“学分”构成。

每个数据流应有良好的命名，它不仅作为数据的标识，还有利于深化对系统的认识。可以考虑使用数据流中最主要的数据项作为该数据流的名称。与加工命名类似地，应该避免使用诸如“数据”、“信息”之类地抽象名词标识数据流。因为加工是一个进行数据处理的黑匣子，因此流入、流出同一加工的数据流应该具有不同的名称。数据流不应包括控制流。数据流反映的是加工处理的对象，控制流是一种选择或用来影响加工的性质，而不是对它进行加工的对象。例如图 3-8 中的“读下张卡片”即属于控制流，不应该在数据流图中画出。

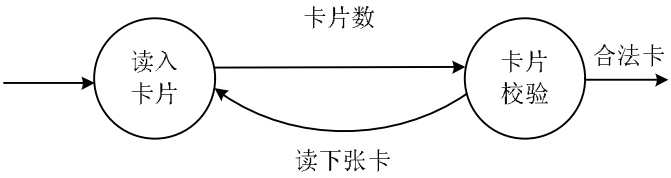


图 3-8　含有控制流的例子

4. 数据存储

数据存储用两条平行线段表示，逻辑上是信息的静态存储。物理上，DFD 中的数据存储可以是计算机系统中的外部或者内部文件、文件的一部分、数据库的元素或记录的一部分等，还可以是一个人工系统中的表册、账单等。数据存储是系统的重要组成部分，在分层数据流图中，通常是局部于某一分解层次的。数据存储可用名词或名词性短语命名，还需要在数据词典中说明其逻辑或者

物理组织要求以及存储介质等。一个数据流从加工流向数据存储，表示该加工对文件写；如果数据流是从数据存储流向加工，表示该加工对文件读。如果加工到数据存储之间的数据流是双向的，表示该加工对文件的操作包括读、写和修改。流入、流出数据存储的数据流的名字通常和数据存储同名，可以不标识。

3.3.3.2 *数据流与加工之间的关系*

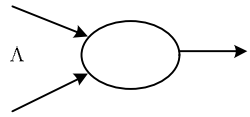
在数据流图中，两个加工之间可以有多个数据流。如果有两个以上的数据流指向同一加工，或一个加工流出两个以上数据流，则这些数据流之间往往存在一定的关系。为表达数据流之间的逻辑联系，可以附加说明标记符号。常见的说明符号及含义如图 3-9 所示。

图 3-9（a）表示流入加工的两个数据流必须同时到达，该加工才能启动。

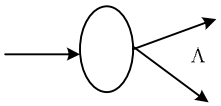
图 3-9（e）表示流入加工的两个数据流只要有一个到达，该加工即可启动。

图 3-9（c）表示流入加工的两个数据流不能同时到达。

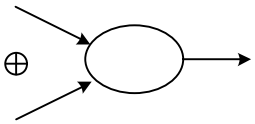
对于图 3-9（b）、图 3-9（d）、图 3-9（f）则是对流出加工的数据流相应关系的对应解释。



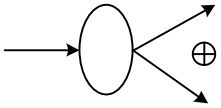
（a）流入加工的两个数据流的联系为与



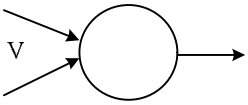
（b）流出加工的两个数据流的联系为与



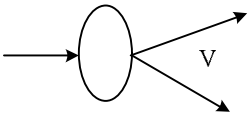
（c）流入加工的两个数据流的联系为异或



（d）流出加工的两个数据流的联系为异或



（e）流入加工的两个数据流的联系为或



（f）流出加工的两个数据流的联系为或

图 3-9　流入/流出同一加工数据流

3.3.3.3 *数据流图的分层*

对于一个大型软件系统，不可能将全部最终的加工和数据流都在一张图上表现出来。这样图面太大，关系复杂，难以理解。结构化控制复杂性的方法，是采用分层技术，用一套分层 DFD 来分解复杂性。分层体现了抽象和信息隐蔽，即上层不考虑下层的细节，暂时掩盖了下层加工的功能及它们的复杂关系。

一个软件系统的一套分层 DFD 图包括顶层 DFD、中间层 DFD 和底层 DFD 组成。顶层图只有一张，它描绘了整个系统的作用范围，可将整个系统作为一个加工，其加工名就是系统名，输入和输出就是系统的所有输入和输出数据，也就是系统与外界的接口；中间层的 DFD 图是对上层父图的分解，它的每一加工还可继续分解细化。当分解一直进行到每个加工的功能独立，简单明确，数据流被严格定义时，即得一组底层 DFD 图。每张底层 DFD 图是由一些不能再分解的加工和简单数

据流组成，这些加工被称之为基本加工。显然，这种自顶向下，逐层地理解和表达系统的分层 DFD 图，是一个控制复杂度，保证分析质量的很好的系统分析方法。

分解示意图如图 3-10 所示。

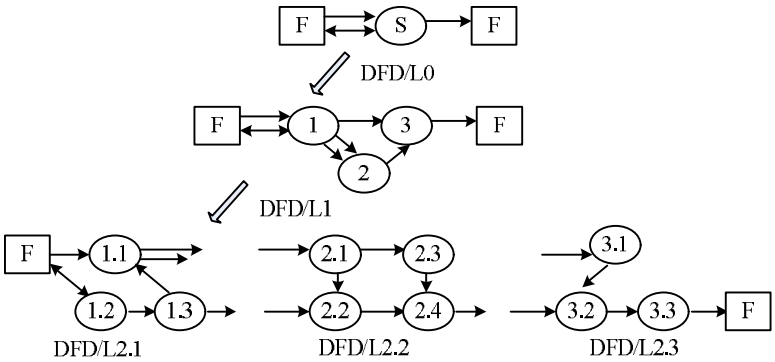


图 3-10　分层数据流图

为了控制分解过程，严格地表现一套 DFD 图，引入如下概念：

（1）父图与子图。如图 3-11 所示的顶层图是整个系统的抽象表示，如果系统 S 可分解为三个子加工 S1、S2、S3，画出相关的数据流，得到顶层下的第一层数据流图。继续分解 S1、S2、S3，得到多层的数据流图。其中上一层图是其分解的下一层图的父图。相反下一层图是上层图的子图。父图里的加工是其相应子图的抽象表示，子图则是其父图中相应加工的细化。

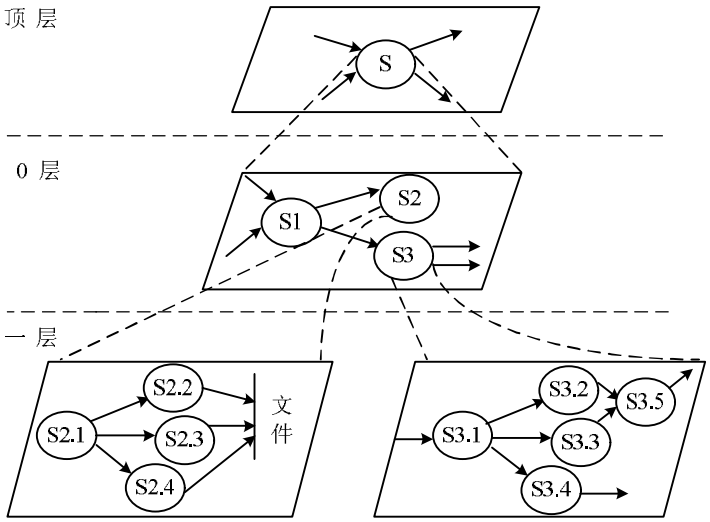


图 3-11　分解示意图

（2）分层图编号。顶层图中加工，即整个系统的编号以 0 表示，以下每一张子图的编号就是上图中相应加工的编号，而子图中每个子加工的编号应该是子图号，加小数点，加局部加工号。在一张分层图中，每个加工的编号中所含小数点的个数，就是该图的层次数。图 3-11 显示了这一编号规则。

（3）父图和子图数据流的平衡性。在绘制分层 DFD 图时，必须注意子图与其父图中相应加工的数据接口保持平衡，即父图中流入流出某加工的数据流，与分解后子图中流入流出加工的数据流

应保持逻辑上的一致。控制分层 DFD 图的数据流的平衡性，是消除 DFD 图错误的一个重要手段。例如图 3-12（b）是图 3-12（a）中 S3 的分解，父图中流入 S3 的数据流为 M、N，流出 S3 的数据流为 O，而在图 3-12（b）中流入的数据流少了 M，显然是错误的。

注意，很多情况下父图和子图数据流的平衡性，不一定是数据流的名称和个数的形式一致，如图 3-12（d）所示，流出的数据流为 J、K，从表面上看，数据流的名字和数量与图 3-12（a）中流入流出 S3 的均不一致，但若数据流 J、K 是随着对图 3-12（a）中加工 S3 的分解，将数据流 O 分解而来的，则图 3-12（d）与图 3-12（a）仍是平衡的。

（4）局部文件。如果某个中间层的 DFD 中的数据存储不是上图中相应加工的外部接口，而只是本图中某些加工之间的信息接口，则称其为局部文件。一个局部文件只有当它作为某个加工的数据接口或某个加工的特定输入或输出时才予以标出。图 3-12 中的数据存储 Fi 就是局部文件。局部文件标出的约定有助于信息隐蔽。

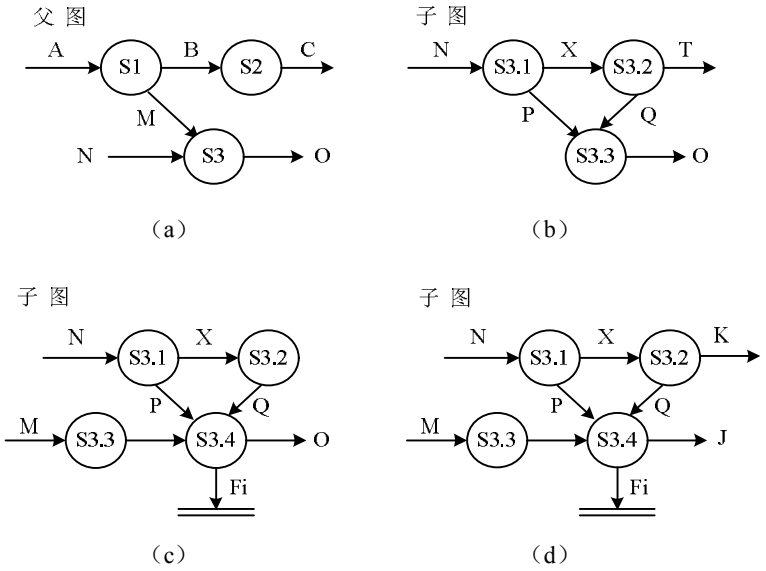


图 3-12　父图和子图的平衡

3.3.3.4　画数据流图的步骤

构造分层 DFD 图的过程就是结构化的系统建模的过程，在对分层数据流图求精过程中，自顶向下逐层地理解和表达系统。经验指出，对不同的软件需要根据不同的思路画 DFD 图，但其基本步骤应是：由外向里，自顶向下，模拟现行的问题处理过程，通过一系列分解步骤，逐步细化、完善求精，最终表现出整个软件系统的构成。具体步骤如下：

（1）找外部实体，确定系统边界，确定数据流源和数据终点。以项目开发计划确定的目标为基础，经过需求获取工作，可以比较容易划定系统的边界，确定系统的数据源点和终点。进而找出外部实体的输入和输出数据流，画出顶层数据流图。

（2）从数据源点出发，按照系统的逻辑需要，逐步画出一系列逻辑加工框，直至数据终点。自顶向下，对每个加工进行内部分解，画出分层数据流图。

（3）按照下述一般原则对数据流图进行复查求精。复查求精应由分析员与用户共同参与，分

析员借助数据流图及数据词典（详见 3.3.4 节），向用户阐述系统输入数据如何一步一步地转变为输出结果。这些阐述集中地反映了分析员当前对于目标系统的认识，用户应认真听取分析员的报告，考察处理是否正确，功能是否完整，并及时纠正与补充。分析员由此对数据流图进一步求精。

构造分层数据流图及求精的一般原则是：

（1）数据流图中所有图形符号必须是四种元素之一，图中的每一个元素都必须有良好的命名。数据流图的主图必须封闭在外部实体之间，实体可以有多个。

需要说明的是，在使用中，为方便起见，还常使用另一套等价的符号，两套符号的对应关系如图 3-13 所示。

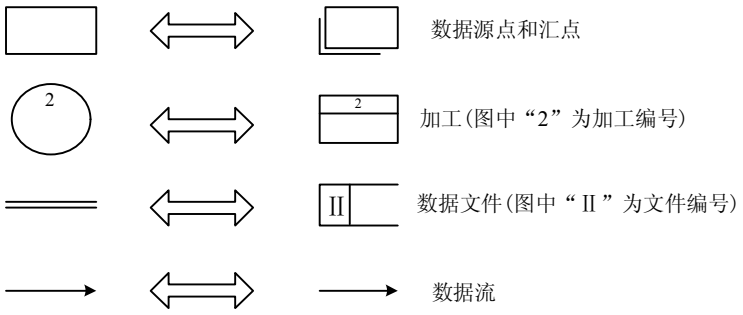


图 3-13　符号对照

（2）分解应该自然，概念上清晰、合理。应使每次分解生成的 DFD 上的数据流和加工都很容易确切地命名，否则说明分解欠合理，应重新考虑。

（3）流入、流出加工的数据流应连续。即每个加工必须有输入、输出数据流，仅有输入（或者仅有输出）数据流的加工是不恰当的。同时流入和流出同一加工的数据流之间应具有对应关系，即每个加工产生的所有数据流都能够由进入该加工的数据流导出。

（4）在分解过程中，注意保持父图和子图数据流的平衡性，合理利用局部文件实现信息隐蔽。

（5）控制单张数据流图的复杂性。数据流图的复杂性体现在一张图上加工和数据流的数目。一个加工的下层子加工应该控制在 7 个以内。流入、流出同一加工的数据流也不应太多。若太多，意味着该加工的功能过于复杂，可能是分解不恰当所至。

（6）当一个加工逻辑足够简单，则分解可以终止。常见的终止加工分解的条件如采用结构化语言对加工进行描述时，不超过一页打印纸；加工只有一个输入数据流和一个输出数据流等。

（7）在软件分析过程中应只考虑稳态，暂时忽略有关细节。如可以暂不考虑初始化、出错路径等细节。数据流图中不应有控制流和控制信息，如图 3-14 所示。其中“X≥0 时”即为控制流。

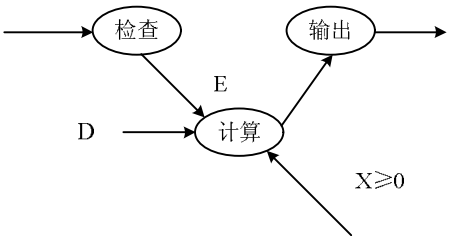


图 3-14　一个带有控制流的错误 DFD

（8）在进行逐层分解时，应注意分解层次的均匀性。避免使一次分解的加工太多和太少。因为太多会使一张图的复杂性增大，太少则会使层次增多，增加了综合评价的工作量。一般应上层分解快些，下层分解慢些。可以将一个系统的全部分层数据流图看作一棵树。顶层加工为树根，所有底层加工为树叶。从树根到所有树叶的路径长度之差不应该太大（理想情况是不超过 2）。两个底层加工的层次相差太大，说明在某个高层加工的分解可能是不适当的，某些加工已是基本加工，而另一些还须加工分解若干次。

3.3.4 数据字典

数据字典是 SA 方法中另一个重要的分析工具。数据流图对信息处理逻辑模型的描述，具有直观、全面、容易理解的优点，但没有准确、完整地定义图中各元素。SA 方法要求对于数据流图中的所有数据流、文件和底层加工进行准确、完整定义。这些图元定义条款汇集在一起即组成数据字典。

在数据字典中给出严格的数据定义可以减少分析人员和用户之间的通信，消除误解，同时它也是以后进行系统设计及维护的重要依据，是进行系统设计的最有价值的文档之一。数据流图和数据字典应该配合使用，数据流图加数据词典构成系统需求分析规格说明的主要部分。

3.3.4.1 数据字典编写的基本要求

（1）对数据流图上的各种元素的定义必须明确、一致且易理解。

（2）命名、编号应与数据流图一致。

（3）对数据流图的成分定义与说明无遗漏，无同名异义，或异名同义。

（4）格式规范，文字精练，符号正确。

3.3.4.2 数据字典的内容和格式

数据词典中包括四类条目：数据流、数据项、文件、基本加工。

（1）数据流条目：数据流条目定义一个数据流的数据项组成。包括名字、别名、编号、来源去向、包含的数据结构名以及处理特点（如使用频率、数据量等）和其他注释信息（如格式、位置等）。

（2）数据项条目：数据项是组成数据流的组成要素，分为基本数据项和结构型数据项。基本数据项是数据处理中基本的不可分割的逻辑单位，如整数、小数、字符串、日期、逻辑值等。在数据词典中通常要求定义基本数据项的逻辑或者物理格式。结构型数据项由若干数据项组成。

（3）文件存储条目：说明存储文件的名称、编号、文件组织方式、记录数及存储介质等。

（4）加工说明条目：说明加工的名称、编号、输入/输出数据流、加工逻辑概括描述（描述方法见 3.3.5 加工逻辑说明）。

为对上述条目进行准确、清晰、无二义性地描述，应对描述的格式进行规范。表 3-1 给出在数据字典的定义式中常用的符号及含义。

3.3.4.3 数据字典的构造及使用

对数据字典中的条目进行定义时，仍体现自顶向下，逐步细化的思想，直至给出基本数据项定义为止。如下例：教学管理系统中的课程管理子系统中的数据文件“学生选课登记表”的格式如表 3-2 所示，在数据字典中的定义格式为：

学生选课登记表= 1{学号+姓名+课程+专业+班级+（备注）}23
学号=“00001”..“99999”
姓名=4{字符}8
课程=课程名+课程类型+学分
专业=“01”..“99”，注：专业代码，两位数字
开课时间=年+月+日
备注=0{字符}40
课程名=“001”..“999”，注：课程代码，三位数字
课程类型=[B | X | R]，注：课程类别一为标识符
B=“必选”
X=“限选”
R=“任选”
学分=“x”

表 3-1　数据字典的定义式中常用的符号及含义

符号	含义	举例
=	被定义为	日期=年+月+日
+	与	学生输入数据=学号+姓名
[···,···][···|···]	或	学生输入数据=[学号|姓名]
{··· ·}	重复	账目={账号+户名+款额+存期+地址}
m{···}n	重复	密码=1{字母+数学}8
(··· ·)	选择	登录信息=用户名+(密码)
“...”	基本数据元素	印密=“0”
..	连接符	学号=“00001”..“99999”

表 3-2　学生选课登记表

学号	姓名	课程	专业	开课时间	备注

数据字典是为分析人员及其他开发人员提供了数据流图中各种元素的详细定义，是需求评审以及后续进行系统设计及维护的重要依据。为了便于使用，也需像普通字典一样，将所定义的所有条目按照一定的编目方法排列起来，并建立索引目录。可以和日常查词典一样，借助于数据词典就可以查出某个名字的具体含义。

3.3.5　加工逻辑说明

在数据流图中，只简单地对每一个加工框进行了编号和命名，没有表达加工的全部内容。为了理解这些基本加工，像数据流条目和文件条目一样，要为每个加工编写详细的加工逻辑说明，这些

说明可以按照加工的编号次序排列，也可按其他确定的次序排列，形成词典的加工条目，与数据流条目和文件条目组成一本完整的数据词典。在将来的设计中，加工逻辑说明就是对应处理程序设计的基础。

一般在数据词典中只列出基本加工的逻辑说明，这是因为随着自顶向下逐层细化，将一个复杂的系统分解成了许多足够简单的基本加工。只要写出每一个基本加工的全部详细逻辑功能，再自底向上综合，就能完成全部逻辑加工。

对数据流图的每一个基本加工的逻辑说明，集中描述一个加工“做什么”，即加工逻辑。加工逻辑是指用户对这个加工的逻辑要求，即这个加工的输出数据流和输入数据流的逻辑关系，描述基本加工是如何把输入数据流变换为输出数据流的加工规则，也包括其他一些与加工有关的信息，如执行条件、优先级、执行频率、出错处理等。注意加工逻辑说明描述的实现加工的策略，而不是实现加工的细节，即不描述具体的加工过程。

加工逻辑说明中包含的信息应是充足的、完备的、有用的、准确、易懂的。目前用于描述加工逻辑说明的工具有结构化英语、判定表和判定树。

3.3.5.1　结构化英语（Structured English）

结构化英语也叫做问题描述语言（Problem Describe Language，PDL），它是在自然语言基础上加了一些限制而得到的一种介于自然语言和形式化语言之间的半形式化语言。它使用有限的词汇和有限的语句来描述加工逻辑。

结构化英语的词汇表由英语命令动词、数据词典中定义的名字、有限的自定义词和逻辑关系词IF_THEN_ELSE、WHILE_DO、REPEAT_UNTIL、CASE_OF 等组成。其动词的含义要具体，不要用抽象的动词。尽可能少用或不用形容词和副词。

结构化自然语言的语法通常分为内外两层。外层语法描述操作的控制结构，如顺序、选择、循环等，这些控制结构将加工中的各个操作连接起来。内层的语法一般没有限制。语言的正文用基本控制结构进行分割，加工中的操作用自然语言短语表示。其基本控制结构有三种：

（1）简单陈述句结构：避免复合语句。

（2）判定结构：IF_THEN_ELSE 或 CASE_OF 结构。

（3）重复结构：WHILE_DO 或 REPEAT_UNTIL 结构。

下面是用结构化语言描述的教学管理系统中，决定学生升留级的加工逻辑说明。

```
BEGIN
  输入学生成绩记录
  REPEAT
    IF 总分≥600 分
    THEN  IF 单科成绩有不及格
                THEN 发升级通知书 AND 发重修单科通知书
                ELSE  发升级通知书
    ELSE  IF 单科成绩有满分
                THEN 发留级通知书 AND 发免修单科通知书
                ELSE  发留级通知书
    UNTIL 学生成绩记录处理结束
END
```

3.3.5.2　判定树（Decision Tree）

在某些数据处理问题中，其数据流图的处理需要依赖于多个逻辑条件的取值，这些取值构成多种不同的情况，满足不同条件，执行相应的不同动作。这类问题适合使用判定树或判定表作为描述加工小说明的工具。

判定树是一种呈树状的图形工具，适合于描述处理中具有多种策略，要根据若干条件的判定，确定所采用策略的情况。

用判定树描述的决定学生升留级的加工逻辑说明如图 3-15 所示。

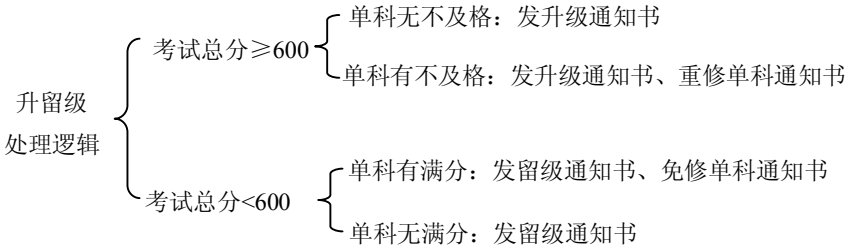


图 3-15　决定学生升留级判定树

判定树具有清晰、直观、易于使用的优点。但当条件多，而且互相组合时，不容易清楚地表达判断过程，因此引入判定表。

3.3.5.3　判定表（Decision Table）

判定表比较适合用来描述数据流图中的加工需要依赖于多个逻辑条件的取值，即完成这一加工的一组动作，是由于某一组条件取值的组合而引发的处理情况。判定表的结构如图 3-16 所示，通常由四部分组成，其中：

（1）基本条件区：列出所有可能的基本判断条件项，通常与次序无关。

（2）基本动作区：列出所有可能采取的动作项，通常与次序无关。

（3）条件组合区：各种条件给出的多种取值，即多个条件所取真假值的组合。

（4）执行动作区：指出在各种条件的特定取值下应采取的动作。在各动作行与条件组合列的交叉处表示在指定条件组合下发生的动作。

图 3-16　判定表的结构

通常将任一条件取值组合及其相应要执行的动作称为规则（在判定表中贯穿条件项和动作项一列）。显然，判定表中列出了多少个条件取值的组合，也就有多少条规则。

上例中升留级处理使用判定表描述如图 3-17 所示。

在实际使用判定表时，常常需要先将其化简。方法是：如果表中有两条或更多的规则具有相同

的动作，并且其条件项之间存在着某种关系，就可设法将它们合并。例如图 3-16 中条件项中的第二条件的取值不同，但规则 1 和规则 3 的动作项是一致的，这表明在第一、第三个条件分别取真值和假值时，不论第二条件取何值，都执行同一动作。这样，可将这两条规则合并，合并后的第二条件取值用“—”表示，以示与取值无关。类似地，若无关条件项“—”在逻辑上又可包含其他的条件项取值，具有相同动作的规则还可进一步合并。

	规则 1	规则 2	规则 3	规则 4	规则 5	规则 6	规则 7	规则 8
考试总分	≥600	≥600	≥600	≥600	<600	<600	<600	<600
单科满分	有	有	无	无	有	有	无	无
单科不及格	有	无	有	无	有	无	有	无
发升级通知书	Y	Y	Y	Y	N	N	N	N
发单科免修通知书	N	N	N	N	Y	Y	N	N
发留级通知书	N	N	N	N	Y	Y	Y	Y
发单科重修通知书	Y	N	Y	N	N	N	N	N

图 3-17　升留级处理逻辑的判定表

按此方法对图 3-17 化简后的判定表如图 3-18 所示。

考试总分	≥600	≥600	<600	<600
单科满分	—	—	有	无
单科不及格	有	无	—	—
发升级通知书	Y	Y	N	N
发单科免修通知书	N	N	Y	N
发留级通知书	N	N	Y	Y
发单科重修通知书	Y	N	N	N

图 3-18　化简后的升留级处理逻辑的判定表

对于需要描述的是由一组操作组成的，其中有些操作是否执行又取决于一组条件，用语言说明难以准确、清楚表达的加工，使用判定表能够把在什么条件下、系统应完成哪些操作表达得十分清楚，准确，一目了然，不容易发生错误和遗漏。但其缺点是描述循环比较困难。有时，判定表可以和结构化英语结合起来使用。

3.4　原型法

目前有代表性的软件需求分析方法除了面向数据流的结构化分析方法外，还有面向数据结构的 Jackson 方法、面向对象的方法和原型法等。由于原型法（Prototyping）改变了传统的系统的分析、

设计和实现三个顺序阶段的观点及自顶向下的开发模式，降低了软件需求的风险，因此该软件开发过程模型可作为获取需求技术并得到广泛的应用。

3.4.1　原型在需求分析中的作用

3.4.1.1　原型的概念

通常，原型是指模拟某种产品的原始模型。在软件开发中，原型（prototype）则是软件的一个早期可运行的版本，它反映最终系统的部分重要功能和特性。是将系统主要功能和接口通过快速开发，制作为“软件样机”，以可视化的形式展现给用户，以征求用户意见，从而明确无误地确定用户需求。同时，原型也可用于征求内部意见，作为分析和设计的接口之一，便于沟通。

3.4.1.2　原型的作用和特点

如前所述，需求分析的难点是构造一个完整准确的规格说明。特别是对于一些大型的软件项目，在开发的早期，当系统分析员与用户一同确定需求时，用户往往对系统只有一个模糊的想法，不能确定真正需要什么，很难完全准确地表达对系统的全面要求。尤其是随着开发工作向前推进，需求还常常会发生变化，同时，对系统运行的效果，只能通过对它的逻辑上的推断，达到各方对系统的共同理解，而不是在实际运行中判断评价。所以尽管采取多种方法，仍很难保证所构造的规格说明能将系统的各个方面都描述的完整、准确、一致，并与实际环境相符。

而原型法可在获得一组基本需求说明后，通过快速分析构造出一个可运行的、满足用户基本要求的软件系统框架。通过运行原型系统，使得用户通过亲身感受得到启发，提出具体的要求和评价。然后开发者根据用户的意见对原型加以改进。随着不断运行原型、评价和修改，获得新的原型版本。如此周而复始，逐步澄清模糊和误解，明确各种需求细节，适应需求的变更，从而提高了最终产品的质量。所以原型法是最准确、最有效的一种需求分析技术。

原型应该具备的第一个特性是“快速”。快速原型的目的是尽快向用户提供一个可在计算机上运行的目标系统的模型，以便使用户和开发者尽可能快地达成对目标系统应该“做什么”的共识。所以，构造原型所做的需求往往是不足的，此时的原型会有某些缺陷，但只要这些缺陷对原型功能的损害不严重，不会使用户对产品的行为产生误解，就可以忽略。

原型应该具备的第二个特性是“易于修改”。如果原型的第一版不是用户所需要的，就必须根据用户的意见迅速地修改它，构建出原型的第二版，以更好地满足用户需求。在实际开发软件产品时，原型的“修改－试用－反馈”过程可能重复多遍，如果修改耗时过多，势必延误软件开发时间。

3.4.1.3　软件原型的分类

虽然软件原型化方法是在研究分析阶段的方法和技术中产生的，但它更针对传统方法所面临的困难。因此，原型化方法也面向软件开发的其他阶段。根据软件项目的特点和运行原型的不同目的，原型有两种不同的类型。

（1）废弃（throwaway）型：先构造一个功能简单而且质量要求不高的模型系统，针对这个模型系统反复进行分析修改，形成比较好的设计思想，据此设计出更加完整、准确、一致、可靠的最终系统。系统构造完成后，原来的模型系统就被废弃。

（2）追加（addon）型：先构造一个功能简单而且质量要求不高的模型系统，作为最终系统的核心，然后通过不断地扩充修改，逐步追加新要求，最后发展成为最终系统。

采用什么形式主要取决于软件项目的特点和开发者的素质，以及支持原型开发的工具和技术，应根据实际情况的特点加以决策。

3.4.1.4 构建原型的方法及工具

原型系统不同于最终系统，它需要快速实现，投入运行。因此，必须注意功能和性能上的取舍。在忽略一切暂时不必关心的部分，快速实现原型时，要能充分地体现原型的作用，满足评价原型的需求。要根据构造原型的目的，明确规定对原型进行考核和评价的内容，如界面形式、系统结构、功能或模拟性能等。构造出来的原型可能是一个忽略了某些细节或功能的整体系统结构，也可以仅仅是一个局部，如用户界面、部分功能算法程序或数据库模式等。总之，在使用原型化方法进行软件开发之前，必须明确使用原型的目的，从而决定分析与构造内容的取舍。对原型的基本要求包括：

（1）体现主要的功能。

（2）提供基本的界面风格。

（3）展示比较模糊的部分，以便于确认或进一步明确，防患于未然。

（4）原型最好是可运行的，至少在各主要功能模块之间能够建立相互连接。

为了快速地构建和修改原型，通常使用下述 3 种方法和工具。

1. 第四代技术

第四代技术包括众多数据库查询和报表语言、程序和应用系统生成器以及其他非常高级的非过程语言。第四代技术使得软件工程师能够快速地生成可执行的代码，因此，它们是较理想的快速原型工具。

2. 可重用的软件构件

另外一种快速构建原型的方法，是使用一组已有的软件构件（也称为组件）来装配（而不是从头构造）原型。软件构件可以是数据结构（或数据库），或软件体系结构构件（即程序），或过程构件（即模块）。必须把软件构件设计成能在不知其内部工作细节的条件下重用。应该注意，现有的软件也可以被用作“新的或改进的”产品的原型。

3. 形式化规格说明和原型环境

在过去的 20 多年中，人们已经研究出许多形式化规格说明语言和工具，用于替代自然语言规格说明技术。今天，形式化语言的倡导者正在开发交互式环境，以便可以调用自动工具把基于形式语言的规格说明翻译成可执行的程序代码，用户能够使用可执行的原型代码去进一步精化形式化的规格说明。

3.4.2 快速原型开发过程

原型的开发和使用过程叫做原型生存期。图 3-19 给出了原型法开发过程，下面分别对过程中的关键步骤进行讨论。

1. 快速分析

在分析者和用户的紧密配合下，快速确定软件系统的基本要求。根据原型所要体现的特性（或界面形式，或处理功能，或总体结构，或模拟性能等），描述基本规格说明，以满足开发原型的需要。快速分析的关键是要注意选取分析和描述的内容，围绕使用原型的目标，集中力量，确定局部的需求说明，从而尽快开始构造原型。

2. 构造原型

在快速分析的基础上，根据基本规格说明，尽快实现一个可运行的系统。为此需要强有力的软件工具的支持，并忽略最终系统在某些细节上的要求，如安全性、健壮性、异常处理等。主要考虑原型系统应充分反映的待评价的特性。

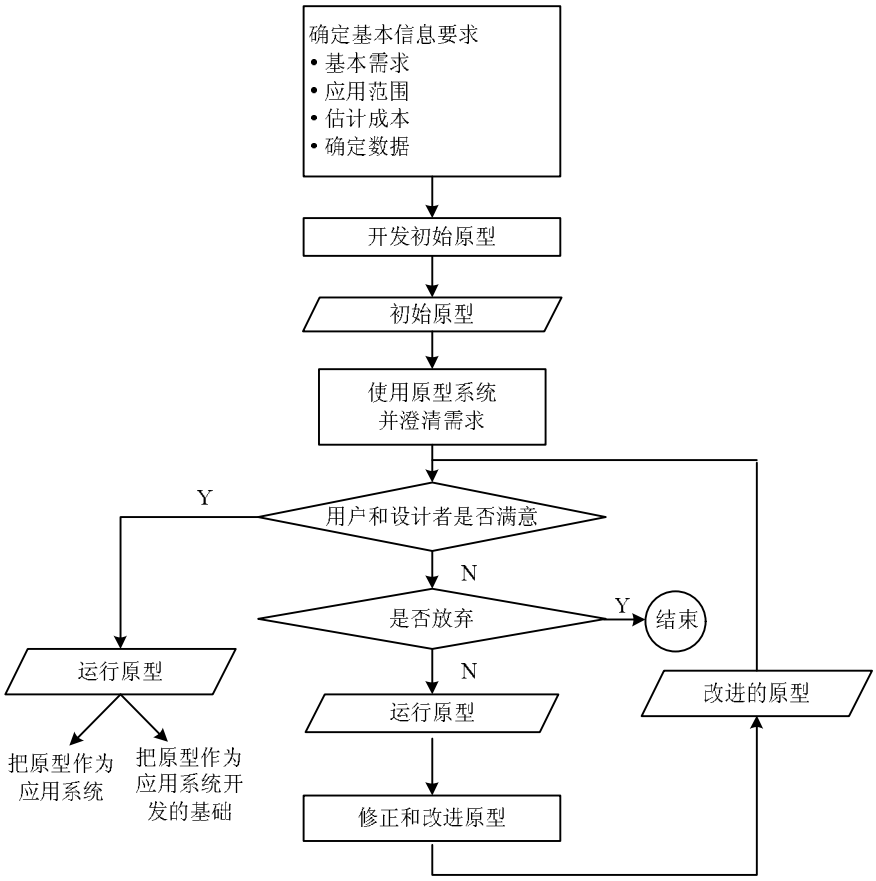


图 3-19　原型法开发过程

提交一个初始原型所需要的时间，根据问题的规模、复杂性、完整程度的不同而不同。一般在 3～6 周提交一个系统的初始原型，最大限度不能超过两个月，两个月后提交的应是一个系统而不是一个原型。

3. 运行和评价原型

这是频繁通信、发现问题、消除误解的重要阶段。由于原型忽略了许多内容，它集中反映了要评价的特性，外观看起来可能会有些残缺不全。用户要在开发者的指导下试用原型，在试用的过程中考核评价原型的特性，分析其运行结果是否满足规格说明的要求，以及规格说明的描述是否满足用户的愿望。纠正误解和错误，增补新的要求，并为满足因环境变化或用户的新设想而引起系统需求的变动并提出全面的修改意见。

4. 修正和改进

根据修改意见进行修改。大多数原型不合适的部分可以修正，使之成为新模型的基础。如果是由于规格说明不准确（有多义性或者未反映用户要求）、不完整（有遗漏）、不一致，或者需求有所变动或增加，则首先要修改并确定规格说明，然后再重新构造或修改原型。但若出现因为严重的理解错误而使正常操作的原型与用户要求相违背时，有可能会产生废品，应当立即放弃，而不能再凑合。

如果用修改原型的过程代替快速分析，就形成了原型开发的迭代过程。开发者和用户在一次次的迭代过程中不断将原型完善，以接近系统的最终要求。

在修改原型的过程中会产生各种各样的积极的或消极的影响，为了控制这些影响，应当有一个词典，用以定义应用的信息流以及各个系统成分之间的关系。另外，在用户积极参与的情况下，保留改进前后的两个原型，一旦用户需要时可以退回，而且贯穿地演示两个可供选择的对象，有助于决策。

5. 判定原型完成

经过修改或改进原型，达到参与者一致认可，则原型开发的迭代过程可以结束。为此，应判断有关应用的实质是否已经掌握，迭代周期是否可以结束等。判定的结果有两个不同的转向，一是继续迭代验证，一是进行详细说明。

6. 判断原型细部是否说明

判断组成原型的细部是否需要严格地加以说明。原型化方法允许对系统必要成分进行严格地详细地说明，例如将需求转化为报表，给出统计数字等。

7. 原型细部的说明

必要时，对那些不能通过模型进行说明的成分，如系统的输入、输出、加工，系统的逻辑功能、数据库组织、系统的可靠性、用户地位等，通过文件加以说明，并利用屏幕进行讨论和确定。严格说明的成分要作为原型化方法的模型编写词典，以得到一个统一的连贯的规格说明提供给开发过程。

8. 判定原型效果

考察用户新加入的需求信息和细部说明信息，看其对模型效果产生的影响，判断是否会影响模块的有效性，如果模型效果受到影响，甚至导致模型失效，则要进行修正和改进。

9. 整理原型和提供文档

整理原型的目的是为进一步开发提供依据。原型的初期需求模型是一个自动的文档。

总之，利用原型化技术，可为软件的开发提供一种完整的、灵活的、近似动态的规格说明方法。

3.5 小结

需求分析阶段的任务就是实现需求工程，包括需求开发和需求管理两部分。本章介绍了需求分析的发现、求精、建模、规格说明和复审的过程，及需求管理的内容；讨论了获取需求的方法及需求分析的原则。

传统软件工程方法学使用结构化分析技术，完成需求分析的工作。结构化分析方法从可行性研究阶段得到的数据流图出发，在用户的协助下，面向数据流自顶向下逐步求精，最终构造出系统的逻辑模型，该模型包括一套分层数据流图，并附加数据字典和加工逻辑说明。数据字典描述在数据流图中出现的所有数据对象及控制信息的特性，并给出它们的准确定义。

原型法改变了系统的分析、设计和实现三个顺序阶段的观点，改变了传统的自顶向下的开发模式，降低了软件需求的风险，因此得到了广泛的应用。实践表明，快速建立软件原型是最准确、最有效和最强大的需求分析技术。快速原型应该具备的基本特性是“快速”和“容易修改”，因此，必须用适当的软件工具支持快速原型技术。通常使用第四代技术、可重用的软件构件、形式化规格

说明与原型环境，快速地构建和修改原型。

在需求分析阶段还应该写出软件需求规格说明书，经过严格评审并得到用户确认之后，作为这个阶段的最终成果。通常主要从一致性、完整性、现实性和有效性等 4 个方面复审软件需求规格说明书。

习题 3

1．为什么可行性研究代替不了需求分析？

2．需求分析阶段的出发点和最终结果各是什么？

3．为什么要进行需求分析？通常对软件系统有哪些需求？

4．怎样与用户有效地沟通以获取用户的真实需求？

5．数据流图和程序流程图是哪些阶段采用的图形工具？它们有何不同？

6．某储蓄所的存、取款业务需求是：储户将填好的存（取）款单及存折交业务员，业务员进行分类处理。如存折不符合规定或填写有误，将存折及存（取）款单返还储户；如果是存款，将存折及存款单、现金交存款业务员处理，存款业务员取出底账登记后，将存折还给储户；如果是取款，将存折及取款单交取款业务员处理，取款业务员取出底账登记后，将存折与现金交付储户。试画出该系统的顶层和 0 层数据流图。

7．画出习题 2 中第 4、第 5 题的数据流图。

8．某考务中心准备开发一个“自学考试考务管理系统”，经调研该系统功能如下：

（1）对考生填写的报名单进行审查，对合格的新生，编好准考证发给他，汇总后的报名单送给阅卷站。

（2）给合格的考生制作考试通知单，将考试科目、时间、地点安排告诉考生。

（3）对阅卷站送来的成绩进行登记，按当年标准审查单科合格者，并发成绩单，对所考专业各科成绩均合格者，发放大专毕业证。

（4）对成绩进行分类统计，产生统计报表。

（5）考生可按准考证号随时查询各科成绩。

请按结构化分析方法进行分析，并画出 DFD（细化至三层），构造数据字典。

9．简述原型法的特点。

10．简述结构化分析方法的步骤。

11．邮寄包裹收费标准如下：

若收件地点在 1000 千米以内，普通件每千克 2 元，挂号件每千克 3 元。若收件地点在 1000 千米以外，普通件每千克 2.5 元，挂号件每千克 3.5 元；若重量大于 30 千克，超出部分每千克加收 0.5 元。请绘制确定收费的判定树和判定表。