第5章 软件工程基础知识

软件工程的概念是在1968年正式提出的，到现在出现了大量的研究成果，也进行了大量的技术实践。软件工程正在逐步发展为一门成熟的专业学科，以解决软件生产的质量和效率问题为宗旨，在软件产业的发展中起到了重要的技术保障和促进作用。

5.1 软件工程基本知识

5.1.1 软件的发展

在计算机发展的早期阶段，大多数人把软件看成是不需预先计划的事情。计算机编程很简单，没有什么系统化的方法。软件的开发没有任何管理，一旦计划延迟了或成本提高了，程序员才开始手忙脚乱地弥补，而他们的努力一般情况下也会取得成功。

计算机系统发展的第二阶段跨越了从20世纪60年代中期到20世纪70年代末期的十余年。多道程序设计、多用户系统引入了人机交互的新概念。但是存在一系列软件相关的问题，当用户需求发生变化时需要修改；当硬件环境更新时需要适应。这些活动统称为软件维护。在软件维护上所花费的精力开始以惊人的速度消耗资源。而且，许多程序的个人化特性使得它们根本不能维护。为了寻找解决的办法，人们开始研究开发技术、思想、方法、工具等问题。软件工程就是用工程、科学和数学的原理来研制、维护计算机软件的有关技术与管理方法。

现在，软件既是一种产品，同时又是开发和运行产品的载体。作为一种产品，它表达了由计算机硬件体现的计算潜能，它就是一个信息转换器——产生、管理、获取、修改、显示或转换信息，这些信息可以很简单，也可以很复杂。作为开发运行产品的载体，软件是计算机控制(操作系统)的基础、信息通信的基础，也是创建和控制其他程序的基础。

5.1.2 软件定义与软件特点

计算机软件是计算机系统中与硬件相互依存的另一部分，是包括程序、数据及相关文档的完整集合。其中，程序是软件开发人员根据用户需求开发的用程序设计语言描述的、

适合计算机执行的指令(语句)序列。数据是使程序能正常操纵信息的数据结构。文档是与程序开发、维护和使用有关的图文资料。

国标(GB)中对计算机软件的定义为:“与计算机系统的操作有关的计算机程序、规程、规则,以及可能有的文件、文档及数据”。

软件在开发、生产、维护和使用等方面与计算机硬件相比存在明显的差异,深入理解软件的定义需要了解软件的特点。软件的特点如下。

(1) 软件是一种逻辑实体,而不是物理实体,具有抽象性。

(2) 软件的生产与硬件不同,没有明显的制作过程,可以大量复制。

(3) 软件在运行、使用期间不存在磨损、老化问题。

(4) 软件的开发运行对计算机系统具有依赖性,这导致了软件移植的问题。

(5) 软件复杂性高,成本昂贵。

(6) 软件开发涉及诸多的社会因素。

5.1.3 软件危机与软件工程

软件危机是指在计算机软件开发和维护过程中所遇到的严重问题。实际上,几乎所有的软件都不同程度地存在这些问题。软件危机主要表现在如下几个方面。

(1) 软件需求的增长得不到满足,用户对系统不满意的情况经常发生。

(2) 软件开发成本和进度无法控制。

(3) 软件质量难以保证。

(4) 软件不可维护或维护程度非常低。

(5) 软件的成本不断提高。

(6) 软件开发生产率的提高不及硬件的发展和应用需求的增长。

总之,可以将软件危机归结为成本、质量、生产率等问题。为了解决软件危机,科学家提出了软件工程的思想。

关于软件工程的定义,国标(GB)中指出:“软件工程是应用于计算机软件的定义、开发和维护的一整套方法、工具、文档、实践标准的工序”。

1993 年 IEEE(Institute of Electrical &Electronic Engineers,电气和电子工程师学会)给出了一个更加综合的定义:“将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程,即将工程化应用于软件中”。

软件工程包括三个要素,即方法、工具和过程。方法是完成软件工程项目的技术手段,工具支持软件的开发、管理、文档生成,过程支持软件开发各个环节的控制、管理。

软件工程的核心思想是把软件产品看作是一个工程产品来处理。

5.1.4 软件工程过程与软件生命周期

1. 软件工程过程

ISO 9000 定义:“软件工程过程(software engineering process)是把输入转化为输出的一

组彼此相关的资源和活动”。

ISO 9000 对软件工程过程的定义支持了它两方面内涵。

其一，软件工程过程是指为获得软件产品，在软件工具支持下由软件工程师完成的一系列软件工程活动。基于这个方面，软件工程过程通常包含 4 种基本活动。

(1) P(Plan)——软件规格说明。规定软件的功能及其运行时的限制。

(2) D(Do)——软件开发。产生满足规格说明的软件。

(3) C(Check)——软件确认。确认软件能够满足客户提出的要求。

(4) A (Action)——软件演进。为满足客户变更要求，软件须在使用的过程中演进。

通常把用户的要求转变成软件产品的过程也叫作软件开发过程。此过程包括对用户的要求进行分析，解释成软件需求，把需求变换成设计，把设计用代码来实现并进行代码测试，有些软件还需要进行代码安装和交付运行。

其二，从软件开发观点来看，使用适当资源(包括人员、硬软件工具、时间等)，为开发软件进行的一组开发活动在过程结束时将输入(用户要求)转化为输出(软件产品)。

所以，软件工程的过程是将软件工程的方法和工具综合起来，以达到合理、及时地进行计算机软件开发的目的。软件工程过程应确定方法使用的顺序、要求交付的文档资料、为保证质量和适应变化所需要的管理、软件开发各个阶段完成的任务。

2. 软件生命周期

通常，将软件产品从提出、实现、使用维护到停止使用退役的过程称为软件生命周期，如图 5-1 所示。

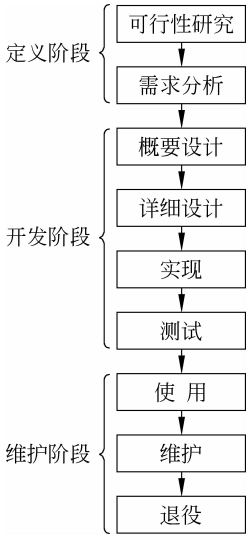

图 5-1　软件生命周期

软件生命周期的主要活动阶段如下。

(1) 可行性研究与计划制订。确定待开发软件系统的开发目标和总的要求，给出它的功能、性能、可靠性以及接口等方面的可能方案，制订完成开发任务的实施计划。

(2) 需求分析。对待开发软件提出的需求进行分析并给出详细定义，编写软件规格说明书及初步的用户手册，提交评审。

(3) 软件设计。系统设计人员和程序设计人员应该在反复理解软件需求的基础上，给出软件的结构、模块和划分、功能的分配及处理流程。在系统比软件复杂的情况下，设计阶段可分解成概要设计阶段和详细设计阶段。编写概要设计说明书、详细设计说明书和测试计划初稿，提交评审。

(4) 软件实现。把软件设计转换成计算机可以接受的程序代码，即完成源程序的编码，编写用户手册、操作手册等面向用户的文档，编写单元测试计划。

(5) 软件测试。设计测试用例，检验软件的各个组成部分。编写测试分析报告。

(6) 运行和维护。将已交付的软件投入运行，并在运行使用中不断地维护，根据新提出的需求进行必要而且可能的扩充和删改。

5.1.5 软件工程的目标与原则

1. 软件工程的目标

软件工程的目标是在给定成本与进度的前提下，开发出满足用户需求且其有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性较好的软件产品。

为了达到软件工程的目标，软件工程主要致力于两个方面的理论和技术性研究：软件开发技术和软件工程管理。

1）软件开发技术

软件开发技术包括软件开发方法学、软件开发过程、软件开发工具和软件工程环境。

其主体内容是软件开发方法学。软件开发方法学根据不同的软件类型，按照不同的观点和原则，对软件开发中应遵循的策略、原则、步骤和必须产生的文档资料都做出了规定，使得软件的开发能够进入规范化和工程化阶段，能够克服早期的手工方法生产中的随意性和非规范性做法。

2）软件工程管理

软件工程管理包括软件管理学、软件工程经济学和软件心理学等内容。

软件工程管理是软件技工程化生产中的重要环节，它要求按照预先制订的计划、进度和预算执行，以实现预期的经济效益和社会效益。多数软件开发项目的失败，是由于管理不当造成的，软件管理学包括人员组织、进度安排、质量保证、配置管理、项目计划等。

软件工程经济学是研究软件开发中成本的估算、成本效益分析的方法和技术，用经济学的基本原理来研究软件工程开发中的经济效益问题。

软件心理学是软件工程领域具有挑战性的一个全新的研究视角，它是从个体心理、人类行为、组织行为和企业文化等角度来研究软件管理和软件工程的。

2. 软件工程的原则

为了达到上述软件工程目标，在软件开发过程中，必须遵循软件工程的基本原则。

（1）抽象：即抽取事物最基本的特性和行为而忽略其非本质的细节。在实施过程中，采用分层次抽象、自顶向下、逐层细化的方法来化解软件开发过程中的复杂性。

（2）信息隐蔽：采用封装技术，将程序模块的实现细节隐藏起来，并提供尽可能简单的模块接口，以便于和其他模块接装在一起。

（3）模块化：模块是程序中相对独立的成分，一个模块是一个独立的编程单位。模块应具有良好的接口定义，大小要适当。

（4）局部化：在一个物理模块内集中逻辑上相互关联的计算资源，保证模块之间具有松散的耦合关系，而模块内部则有较强的内聚性，有助于控制系统的复杂性。

（5）确定性：软件开发过程中所有概念的表达应该是确定的、无歧义的、规范的。

（6）一致性：在程序、数据和文档的整个软件系统的各模块中，应使用已知的概念、

符号和术语；程序的内部和外部接口应保持一致，系统规格说明与系统行为应保持一致。

(7) 完备性：指软件系统不丢失任何重要成分，完全实现系统所需的功能。

(8) 可验证性：开发大型软件系统需要对系统自顶向下，逐层分解。系统分解应遵循易于检查、测评、评审的原则，以确保系统的正确性。

5.1.6 软件开发工具与软件开发环境

方法和工具是软件工程学中同一个问题的两个不同方面，方法是工具研制的先导，工具是方法的实在体现。软件工程方法的研究成果最终为软件工具和系统。软件工程环境就是围绕着软件开发的一定目标而组织在一起的一组相关软件工具的有机集合。

1. 软件开发工具

早期的工具主要支持软件生存周期的后期阶段的开发，如编码和调试，这一时期的编程工作量大，质量和进度难以保证，人们将很多精力和时间花费在程序的编制和调试上，而在更重要的软件的需求和设计上反而没有必要的精力和时间的投入。

软件开发工具的发展和完善将促进软件开发方法的进步和完善，促成软件开发的高速度和高质量。软件开发工具的发展是从单项工具的开发逐步向集成工具发展的，软件开发工具为软件工程方法提供了自动的或半自动的软件支撑环境。同时，软件开发方法的有效应用也必须得到软件工具的支持，否则方法将难以有效地实施。

2. 软件开发环境

软件开发环境(或称软件工程环境)是全面支持软件开发全过程的软件工具集合，它们按照一定的方法或模式组合在一起，支持软件生命周期内各个阶段中各项任务的完成。

计算机辅助软件工程(Computer Aided Software Engineering，CASE)是当前软件开发环境中富有特色的研究工作和发展方向。CASE 将各种软件工具、开发机器和一个存放开发过程信息的中心数据库组合起来，形成软件工程环境。CASE 的成功产品将最大限度地降低软件开发的技术难度，并使软件开发的质量得到保证。

5.2 结构化分析方法

软件开发方法是软件开发过程中所遵循的方法和步骤，研究和使用软件开发方法的目的在于有效地得到某些满足质量要求的软件产品，即程序和文档。软件开发方法包括分析方法、设计方法和程序设计方法。

结构化方法是指根据某种原理，使用一定的工具，遵循的是自顶向下、逐步求精、模块化的原则，按照特定步骤工作的软件开发方法。它实际上由三部分构成的：结构化分析(Structured Analysis，SA)、结构化设计(Structured Design，SD)和结构化程序设计(Structured Programming，SP)，其核心和基础是结构化程序设计理论。

5.2.1 需求分析与需求分析方法

软件需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望和要求。需求分析的任务是发现需求(搞清楚用户需要用软件解决什么问题)、求精、建模和定义需求。需求分析将创建所需的数据模型、功能模型和控制模型。

1. 需求分析的定义

需求分析是指开发人员要进行细致的调查分析以便准确理解用户的要求,将用户的需求陈述转化为完整的需求定义,再由需求定义转换到相应的需求格式说明的过程。1997 年,IEEE 软件工程标准词汇表中的需求分析定义如下。

(1) 用户解决问题或达到目标所需的条件或权能。

(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(1)或(2)所描述的条件或权能的文档说明。

由需求分析的定义可知,需求分析的内容包括提炼、分析和仔细审查已收集到的需求,确保所有利益相关者都明白其含义并找出其中的错误、遗漏或其他不足的地方,从用户最初的非形式化需求到满足用户对软件产品的要求的映射,对用户意图不断进行提示和判断。

2. 需求分析阶段的工作

需求分析阶段的工作,可以概括为以下 4 个方面。

(1) 需求获取。需求获取的目的是确定对目标系统的各方面需求,涉及的主要任务是建立获取用户需求的方法框架,并支持和监控需求获取的过程。

在需求获取过程中,要在与用户交流的过程中不断收集用户的各种信息,通过认真理解用户的各种需求,澄清某些模糊的需求,去除不合理的需求,全面地提炼出系统的功能性需求与非功能性需求。一般地,功能性与非功能性需求包括系统功能、物理环境、用户界面、用户因素、资源、安全性、质量保证以及其他约束。

(2) 需求分析。对获取的需求进行分析和综合,最终给出系统的解决方案和目标系统的逻辑模型。

(3) 编写需求规格说明书。需求规格说明书作为需求分析的阶段成果,可为用户、分析人员和设计人员之间的交流提供方便,可直接支持目标软件系统的确认,还可作为控制软件开发进程的依据。

(4) 需求评审。在需求分析的最后一步,要对需求分析阶段的工作进行复审,验证需求文档的一致性、可行性、完整性和有效性。

3. 需求分析方法

常见的需求分析方法有以下两种。

(1) 结构化分析方法：主要包括面向数据流的结构化分析方法(Structured Analysis，SA)、面向数据结构的 Jackson 方法(Jackson System Development Method，JSD)以及面向数据结构的结构化数据系统开发方法(Data Structured System Development Method，DSSD)。

(2) 面向对象的分析方法(Object-Oriented Analysis，OOA)：从需求分析建立的模型的特性来分，又分为表态分析方法和动态分析方法。

5.2.2 结构化分析方法

1. 结构化分析方法

结构化分析方法是结构化程序设计理论在软件需求分析阶段的运用。它起源于 20 世纪 70 年代的基于功能分解的分析方法，可帮助我们弄清楚用户对软件的需求。

对于面向数据流的结构化分析方法，按照 DeMaro 的定义："结构化分析就是使用数据流图、数据字典、结构化英语、判定表和判定树等工具，来建立一种新的、称为结构化规格说明的目标文档。"

结构化分析方法着眼于数据流自顶向下、逐层分解、模块化的思想建立系统的处理流程，以数据流图和数据字典为主要工具建立系统的逻辑模型。

结构化分析的步骤如下。

(1) 通过对用户的调查，以软件需求为线索，获得当前系统的具体模型。

(2) 去掉具体模型中的非本质因素，抽象出当前系统的逻辑模型。

(3) 根据计算机的特点分析当前系统与目标系统的差别，建立目标系统的逻辑模型。

(4) 完善目标系统并补充细节，写出目标系统的软件需求规格说明。

(5) 评审直到确认完全符合用户对软件的需求。

2. 结构化分析的常用工具

结构化分析的常用工具有数据流图、数据字典、判定表和判定树等。

1) 数据流图

数据流图(Data Flow Diagram，DFD)是结构化分析方法中用于表示系统逻辑模型的一种工具。它以图形的方式描绘数据在系统中流动和处理的过程，是需求理解的逻辑模型的图形表示。由于它只反映系统必须完成的逻辑功能，所以是一种功能模型，直接支持系统的功能建模。

数据流图从数据传递和加工的角度来刻画数据流从输入到输出的移动变换过程。数据流图中的主要图形元素如图 5-2 所示。

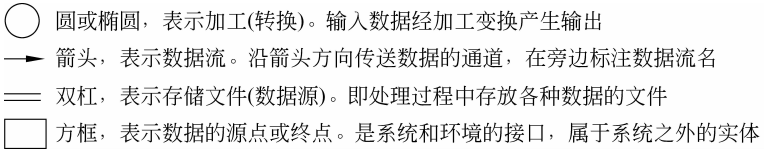

图 5-2 数据流图中的主要图形元素

一般地，通过对实际系统的了解和分析后，使用数据流图为系统建立逻辑模型。建立数据流图的步骤如下。

（1）由外向里：先画系统的输入输出，然后画系统内部。

（2）自顶向下：顺序完成顶层、中间层、底层数据流图。

（3）逐层分解。

数据流图的建立是从顶层开始的，顶层的数据流图（如图 5-3 所示）包含所有相关的外部实体以及它们与软件中间的数据流，其作用主要是描述软件的作用范围，对总体功能、输入输出进行抽象描述，并反映软件与系统、环境的关系。

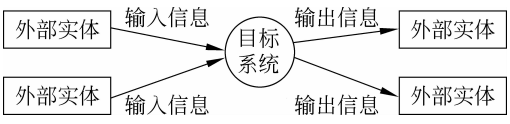

图 5-3　顶层数据流图

为了表达数据处理过程中的数据加工情况，用一个数据流图往往是不够的。稍微复杂的实际问题，在数据流图上常常出现十几个甚至几十个加工。这样的数据流图看起来很不清楚。层次结构的数据流图能很好地解决这一问题。按照系统的层次结构进行逐步分解，并以分层的数据流图反映这种结构关系，能使整个系统清楚地表达和容易理解。

为使所构造的数据流图能够完整、准确、规范地表达系统的数据处理过程，应遵循以下数据流图的构造规则和注意事项。

（1）对每个加工处理建立唯一、层次性的编号，要求每个加工处理既有输入又有输出。

（2）数据存储之间不应该有数据流。

（3）数据流图的一致性包括数据守恒和数据存储文件的使用。其中有两种不遵守数据守恒规则的情况：

① 某个处理用来产生输入的数据没有输入，即出现遗漏。

② 某个处理的一些输入并未在处理中使用以产生输出。

数据存储（文件）应为数据流图中的读和写，而不是只读不写或只写不读。

（4）父图、子图关系与平衡规则是指相邻两层数据流图之间具有父、子关系，子图代表父图中某个加工的详细描述，父图表示子图间的接口。子图个数不大于父图中的处理个数。规定任何一个数据流子图必须与它上一层的一个加工对应，两者的输入数据流和输出数据流必须一致。

例如，学生选课的数据流图如图 5-4 所示。

2）数据字典

数据字典（Data Dictionary，DD）是结构化分析方法的核心。是所有与系统相关的数据元素的一个有组织的列表以及精确的、严格的定义，使得用户和系统分析员对于输入输出、存储成分和中间计算结果有共同的理解。数据字典把不同的需求文档和分析模型紧密地结合在一起，与各模型的图形表示配合，能清楚地表达数据处理的要求。

概括地说，数据字典就是用来定义数据流图中的各个成分的具体含义的，它以一种准确的、无二义性的说明方式为系统的分析、设计及维护提供了有关元素的一致的定义和详细的描述。它和数据流图共同构成了系统的逻辑模型，是需求规格说明书的主要组成

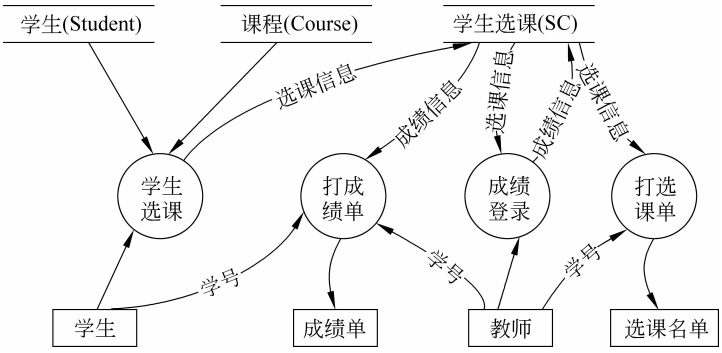

图 5-4　学生选课的数据流图

部分。

在编制数据字典的过程中，常使用定义式方式来描述数据结构，表 5-1 给出了常用的定义式符号及说明。

表 5-1　定义式符号及说明

符　　号	含　　义	例子及说明
=	被定义为	
+	与	$x=a+b$ 表示 x 由 a 和 b 组成
{...|...}	或	$x=\{a\|b\}$ 表示 x 由 a 或 b 组成
$M\{...\}m$	规定次数的重复	$x=2\{a\}5$ 表示 x 中最少出现 2 次 a，最多出现 5 次 a
{...}	重复	$x=\{a\}$ 表示 x 由 0 个或多个 a 组成
(...)	可选	$x=(a)$ 表示 a 可在 x 中出现，也可不出现
..	连接符	$x=1..9$，表示 x 可取 1～9 中任意一个值
* *	注释	

3）判定表

数据流图中某个加工的一组动作有些情况下依赖于多个逻辑条件的取值。这时，用自然语言或结构化语言都不易描述清楚，而用表的形式则一目了然。当数据流图中的加工要依赖于多个逻辑条件的取值，即当完成该加工的一组动作是由于某一组条件取值的组合而引发的时候，使用判定表描述比较合适。

判定表由 4 部分组成包括基本条件、基本动作、条件项和动作项，如图 5-5 所示。

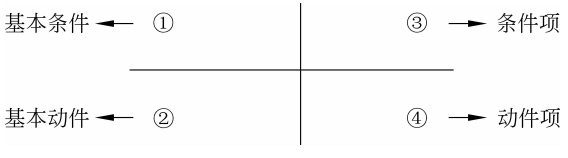

图 5-5　判定表组成

(1) 基本条件：列出了各种可能的条件。

(2) 基本动作：列出了所有的操作。

(3) 条件项：列出了各种可能的条件组合。

(4) 动作项：列出在对应的条件组合下所选的操作。

例如，“检查发货单”的加工逻辑描述如下。

用表 5-2 所示的“检查发货单”判定表来表示各种条件的发货过程。

表 5-2 “检查发货单”判定表

		1	2	3	4
条件	发货单金额	＞$500	＞$500	≤$500	≤$500
	赊欠情况	＞60 天	≤60 天	＞60 天	≤60 天
操作	不发出批准书	√			
	发出批准书		√	√	√
	发出发货单		√	√	√
	发出赊欠报告			√	

4) 判定树

判定树是判定表的变形，一般情况下它比判定表更加直观，且易于理解和使用。

使用判定树进行描述时，应先从问题定义的文字描述中分清哪些是判定的条件，哪些是判定的结论，根据描述问题中的连接词找出判定条件之间的从属关系、并列关系、选择关系，根据它们构造判定树。

如图 5-6 所示为判定树的示例。

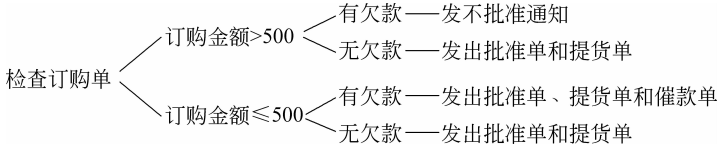

图 5-6 “检查订货单”判定树

5.2.3 软件需求规格说明书

软件需求规格说明书(Software Requirement Specification，SRS)是需求分析阶段的最后成果，是软件开发中的重要文档之一。软件需求规格说明书对所开发的软件的功能、性能、用户界面及运行环境等进行详细的说明。它是在用户与开发人员双方对软件需求取得共同理解并达成协议的条件下编写的，也是实施开发工作的基础。

1. 软件需求规格说明书的作用

(1) 便于用户与开发人员之间的理解和交流。

(2) 反映出用户的问题结构，可作为软件开发工作的基础和依据。