第一篇

软件开发生命周期概念

第 1 章

软件开发生命周期基本概念

本章目标

- 了解软件的定义、特征和分类
- 了解软件产生的发展阶段
- 理解软件危机的表现、原因和解决途径
- 了解软件开发生命周期的含义、基本原理和作用

本章导读

什么是软件呢？难道仅仅就是指我们平时看到的一张光碟、一张磁盘或者是一堆代码吗？很多人都听说过经济危机，那软件危机又是什么，有经济危机那么可怕吗？又该如何解决软件危机呢？同样，很多人也听说过水利工程、电力工程、建筑工程，那软件开发生命周期又是指什么呢？它究竟是一门技术学还是管理学呢？这一连串的疑问将通过本章的学习找到答案。

1.1 软件的基本概念

1.1.1 软件定义

什么是软件呢？这个看似简单的问题，直到 20 世纪 70 年代以后才有了比较科学的认识，而在此之前人们的认识都很片面、模糊甚至荒谬。有人说软件就是一张光碟、一张磁盘，因为他每次买软件其实就是买光碟。这种说法显然是荒谬的，因为光碟、磁盘这些只不过是程序、软件的载体，而并不是软件本身。另一些人似乎高明一些，说软件就是一堆程序，因为平时大家都把软件开发直接说成程序开发。这种说法也是片面的，应该说软件确实离不开程序，但是程序也仅仅是软件这个概念的一部分，而不是全部。

“软件”这一名词是在 20 世纪 60 年代初从国外传来的。由于“software”一词由 soft 和 ware 两部分组成，所以有人译作“软制品”，也有人译作“软体”，现在比较统一的说法叫

“软件”。对于它的一种公认的解释为：软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据和相关文档的完整集合。

- 程序一般认为是能够完成预定功能和性能的可执行的指令。对于面向过程的程序，我们一般把它理解为“算法”结合“数据结构”；而对于面向对象的程序，我们一般把它理解为“对象”结合“消息”。
- 数据一般认为是使程序能够适当地操作信息的数据结构。
- 文档是与程序开发、维护和使用有关的图文材料。文档在软件开发生命周期中特别重要，文档是否规范与齐全，是衡量软件企业是否成熟的重要标志之一。软件文档分为开发文档和管理文档两大类。开发文档主要由项目组书写，用于指导软件开发；管理文档主要由软件开发生命周期管理部门书写，用于指导软件管理和决策。

1.1.2 软件的特征

为了能够全面、正确地理解软件，让我们来看看软件的特点。

(1) 软件是一种逻辑实体，无形态，具有抽象性。人们可以把它记录在纸面上，存储在计算机的硬盘上，又或者刻在光碟上，但纸、硬盘和光碟都只不过是它的载体，而不是软件本身。软件这种无形体只能通过分析、思考和判断去理解它。

(2) 软件的运行和使用期间，没有硬件那样的机械、磨损、老化问题。任何硬件产品一开始并不好使用，随着磨合调整时期一过，产品达到了最佳的使用状态；但随着时间的推移，由于磨损等原因又变得很难使用甚至不可使用，如图 1-1(a)所示。软件则不同，它不存在磨损和老化问题，但它存在退化问题。理想状态，随着软件的不断维护，软件会越来越好用。而实际情况往往是，随着软件的不断维护修改，每次修改都必不可少地引入新的错误，这样一次一次修改，导致软件失效率升高，如图 1-1(b)所示。

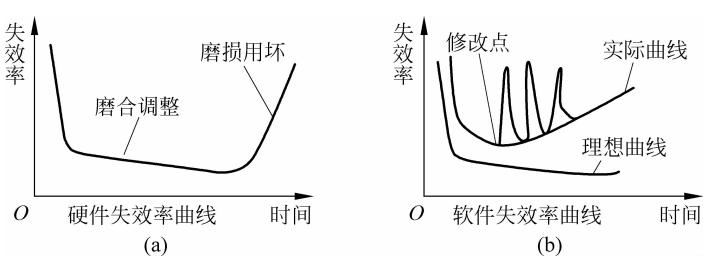

图 1-1 硬件与软件失效率对比

(3) 软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。软件不能完全摆脱硬件单独活动，在开发和运行中必须以硬件提供的条件为依据。有的软件只能在某种型号的计算机上运行，而有的软件只能在某种操作系统上运行。

为了解除这种依赖性，在软件开发中提出了“软件移植”问题，并且把软件的可移植性作为衡量软件质量的因素之一。

(4) 软件的开发至今尚未完全摆脱手工的方式。人们一直期待软件的生产能够像硬件的生产那样实现全自动，将原料放入一个全自动机器，从另一端就可以输出成品。近年来软件技术虽然取得了不少进展，提出了许多新的开发方法，如组件技术、复用技术，也研制了一些高效的开发工具和开发环境，但这种期待软件生产实现全自动的梦想至今未能实现。软件的开发仍然以手工开发方式为主，开发的效率自然受到很大的限制。

(5) 软件是复杂的。开发软件是一种高强度的脑力劳动，没有哪一个软件人员认为这是一项轻松的工作。软件的复杂性主要表现在两个方面。一方面，可能来自程序逻辑结构的复杂性，如像操作系统这样复杂的系统软件，需要处理各种可能的情况，其复杂度是不言而喻的。另一方面，也可能来自它所反映的实际问题的复杂性，如开发一个图书馆管理信息系统，要求软件人员对图书领域的专门知识，诸如图书的编码规则和借还书的流程等有所了解。图 1-2 表明尽管软件技术有了不少的发展，但是发展的速度一直落后于软件需求的发展，而且这种差距越拉越大。

(6) 软件成本相当昂贵。大家知道，很多正版软件居然要卖到几千甚至上万元，似乎不可思议。这主要是因为软件的研制需要投入大量的、复杂的、高强度的脑力劳动，成本自然比较高。从图 1-3 可以看出一个戏剧性的变化趋势。随着时间的推移，花费在软件上的成本所占整个系统成本的比例逐年增加，在 20 世纪 80 年代以后，软件的开销就开始大大超过硬件的开销了，时至今日，这种趋势更为明显。

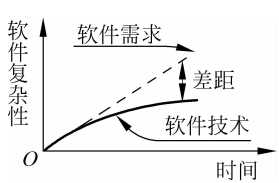

图 1-2　软件技术的发展落后于需求

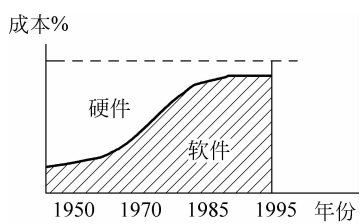

图 1-3　硬件、软件成本的比例变化

1.1.3　软件的分类

大家可能经常听到有人说这是“系统软件”，那是“大型软件”、“定制软件”等，这其实是对软件类型的一种描述。站在不同的角度可以对软件做出不同的分类，就像以身高可以将人分为高人和矮人，以体重可以将人分为胖人和瘦人。以下介绍一些常用的分类方式。

1. 按软件的功能进行划分

(1) 系统软件。能与计算机硬件紧密结合在一起,使计算机系统各个部分、相关的软件和数据协调、高效地工作的软件。例如,操作系统(Windows、Linux、Unix 等)、数据库管理系统(Oracle、SQL Server、DB2 等)、设备驱动程序(网卡驱动程序、显卡驱动程序等)等。

(2) 支撑软件。协助用户开发软件的工具性软件,包括帮助程序人员开发软件产品的工具,也包括帮助管理人员控制开发进程的工具。例如,VS. NET、. NET Framework、JDK、Rational Rose 等。

(3) 应用软件。在特定领域内开发,为特定目的服务的一类软件。例如,用在工业控制上的计算机辅助制造 CAD 软件、系统仿真软件、人工智能方面的模式识别软件等;用在办公事务方面的 Word、Excel、WPS;用在学习方面的"我爱背单词"、"五笔速成"等教学软件;用在娱乐方面的 RealPlay、ACDSee、"魔兽争霸"等软件。

2. 按软件规模进行划分

按开发软件所需的人力、时间以及完成的源程序行数,可确定 6 种不同规模的软件,如表 1-1 所示。

表 1-1 软件规模的分类

类 型	参加人员数	研制期限	产品规模/源程序行数
微型	1	1～4 周	500
小型	1	1～6 月	1000～2000
中型	2～5	1～2 年	5000～50000
大型	5～20	2～3 年	50000～100000
甚大型	100～1000	4～5 年	1000000
极大型	2000～5000	5～10 年	1000000～10000000

3. 按软件服务对象的范围划分

(1) 定制软件。又称项目软件,这类产品受特定的客户委托,由软件公司专门为这类客户开发。例如,某学校为使其图书管理实现信息化,特委托某软件公司根据该校图书馆自身的特点来定制一套图书管理信息系统软件;某电器销售公司为扩大其在市场上的影响力,特委托某软件公司帮其建立网站程序等,都属于定制软件。

(2) 通用软件。又称产品软件,这类软件由软件开发公司根据市场需求开发出来,到市场上公开销售。例如,微软公司看到人们在日常的办公和事务中经常需要输入汉字、绘制一些表格和编辑一些文档等,就开发了 Office 系列办公软件;又如很多游戏软件也属于通用软件。

通俗地讲，定制软件是先有了买家再开发软件；而通用软件是先开发软件再找买家。

1.1.4　软件生产的发展

软件生产的发展大致可以分为三个阶段。

第一个阶段我们把它叫做程序设计时期(1946—1956 年)。这个时期很多人都认为开发软件是一件无需预先计划的事，开发软件就是编写程序。这时的软件往往规模很小，而且往往是自己开发自己使用。由于软件规模小，难度低，开发人员几乎是一上来就编写程序，没有系统化的预先规划，也没有留下什么相关文档，所有编程之外的需求设计等都在大脑中完成。

第二个阶段我们把它叫做程序系统时期(1956—1968 年)。这个时期的软件不再是自己开发自己使用，而是可以作为商品出售。软件的规模和难度都有所增加，原来那种单兵作战的个体开发很难胜任，于是出现了多人分工合作的“作坊式”开发模式。而且人们逐渐认识到了文档的重要性，所以软件除了程序之外还有了一些简单的说明。

第三个阶段我们把它叫做软件开发生命周期时期(1968 年至今)。这个时期的软件已经全面实现了商品化。软件的规模和难度显著增加，人们开始使用软件开发生命周期方法来合理科学地开发软件了。软件开发生命周期学把软件看作程序、数据和相关文档的一个综合体，所以这个时期把文档的重要性提到了空前的高度。

随着软件生产的发展，人们对软件开发的认识也有了一些新的转变。早些时期，人们曾经把程序设计看作一种任人发挥创造才能的技术领域，认为写得越短小精悍、越高深莫测、越多技巧，甚至越让人读不懂的程序才越是高手写出的好程序。因为那时的程序规模和难度都很小，这种“孤芳自赏”的行为暂时倒没有带来多大的危害。但随着程序规模和难度的日益提高，程序中这种难以理解的技巧不再适用，人们要求程序要容易看懂、容易使用，并且要容易修改和扩充。于是，程序便从个人按自己意图创造的“艺术品”转变为能为广大用户接受的工程化产品。

1.2　软件危机

1.2.1　软件危机的表现

大家可能经常听说过“经济危机”，其实在软件领域也有个“软件危机”的说法。

20 世纪 60 年代中后期，软件商品化的出现，像催化剂一样，使计算机应用的广度和深度得到迅速扩大，由此而带来以下的问题。

- 应用领域不断扩大，计算机数量猛增——软件需求量倍增；
- 要求计算机能做更多的事，解决更多的问题——软件规模增大；
- 要求解决问题的深度增加——软件复杂程度增大。

这样，一方面是大家对软件的美好愿望和强烈要求，而另一方面是软件的拙劣表现和

脆弱能力，这样巨大的反差便形成了软件危机。所以软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重的问题，其实质就是软件生产的进度、数量、质量与社会对软件的需求量和希望要求相距甚远，与硬件发展更是形成强烈反差。

从软件危机的定义我们可以看出软件危机主要表现在两方面：一方面是开发，如何开发出满足日益增长的需求的软件；另一方面是维护，如何维护数量不断膨胀的已有软件。

具体来说，软件危机主要有如下一些典型的表现。

1. 软件开发方面

(1) 软件开发进度和成本无法有效控制。例如，一个原计划投入 2 万元用时三个月的软件，结果花了 30 多万元耗时两年多才完成。这种情况时有发生，严重降低了软件公司在人们心目中的信誉度。而为了赶进度和节约成本，一些软件公司所采取的权宜之计，如临时雇用一些对本系统不太熟悉的软件开发新手匆忙的连夜加班追赶进度，这样的做法犹如饮鸩止渴，治标不治本，往往损害了软件的自身质量，从而引起用户更为不满。

(2) 用户对已完成的软件系统不满意的情况时有发生。软件开发人员往往在对用户需求知之甚少的情况下，就仓促上阵匆忙着手编写程序。这就好比一个裁缝在还没有听清楚客户的要求，没有量好客户身材的情况下就匆忙裁剪衣服，势必做出的衣服不是款式不对就是尺寸不合身。

(3) 软件产品的质量往往不可靠。死机、崩溃、出错、数据文件丢失、无法移植等情况时有发生。

2. 软件维护方面

(1) 软件往往是不可维护的。很多程序中的问题是难以改正甚至不可改正的。例如，一些软件在设计之初便忽略了软件的可移植性，当软件需要运行在新的软件环境或硬件环境中时，除了重新编写之外别无他法。

(2) 软件缺乏相应的文档资料。软件除了程序和数据之外还应该包括文档资料。软件管理人员可以使用这些文档资料来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员，这些文档资料更是至关重要。

以上只是列举了软件危机中的一些典型表现，与软件开发和维护有关的危机问题远远不止这些。

1.2.2 软件危机的原因

正如生了病就要找到病因一样，那么产生软件危机的原因是什么呢？按照辩证思维方式，我们应该从客观和主观两方面来查找原因。

1. 客观原因

从客观上来看，有软件本身的原因。软件本身是相当复杂的，而且软件制作是手工劳动，是智力产品，生产率低。软件又是逻辑实体，出错容易，纠错困难。

对于软件的复杂性这一客观事实我们几乎无法改变。

2. 主观原因

出现软件危机的主观原因就是人们在软件的开发和维护方面持有不正确的观念和采用了不正确的方法。这里仅就一些典型的似是而非的论调进行批驳。

错误观点一："开发软件就是编写程序。"

持这种观点的人总是认为去软件公司上班的人都是去编程序，难道软件公司的所有职员都是编程人员吗？难道软件公司每接到一个软件项目，不管三七二十一，一上来就开始编写程序吗？

如果一个软件项目很小，小到诸如仅仅是计算一个三角形的面积这样的小程序，是可以一上来就开始编写程序的，完全没有必要劳师动众地大搞一番需求分析和设计。其实也不是说没有需求分析和设计，而是说由于问题实在太简单了，这些工作可以直接在大脑里面完成。这就好比农村里面谁家要搭猪圈，一般是左邻右舍找几个弟兄，你一块砖我一块砖就开始搭建了，也没见到谁会去法国请个著名建筑师来设计猪圈。

如果一个软件项目比较大，开发软件就不仅仅是上来就编程序这么简单的事了。这就好比盖一栋大楼，不可能一上来就叫上几百个工人你一块砖我一块砖，那样只会陷入一片混乱。

我们必须首先进行问题定义，以明白究竟我们要开发的软件是什么；这就好比盖一栋大楼，我们首先要知道盖的是高层、电梯公寓还是别墅。然后进行可行性分析，以明确软件开发的成本进度以及法律等方面是否可取；这就好比盖一栋大楼，开发商应该事先估算一下自己有没有足够的钱，能否按时交房，所盖大楼是否非法用地等。再接着进行需求分析，以了解用户究竟要求软件公司帮他们开发一个能实现什么功能，达到什么性能的软件；这就好比盖一栋大楼，必须要搞清楚到底要盖的小户型居多还是大户型居多，住户希望使用中央空调还是独立空调的排气结构等。之后就应该是软件的设计，包括概要设计和详细设计；这就好比盖一栋大楼，我们应该设计相应的设计图纸。然后才轮到编写程序，将设计转换为代码；这就好比盖一栋大楼，民工们实际一砖一瓦地堆砌。这以后还要进行软件测试，对编好的软件进行检查；这就好比盖一栋大楼，楼封顶了之后要找相关的检测部门来验收房屋的质量。

最后，软件在交付用户使用后不是撒手不管了，而是需要长期维护；这就好比盖一栋大楼，住户住进去之后，开发商还应该对房屋可能会出现的漏水、裂缝等质量问题进行维护。

以上我们通过一个盖大楼的例子对开发软件的一个生命周期作了一个完整的描述，可见开发软件并非仅仅就是编写程序，事实上编写程序仅仅是开发软件工作的一部分，甚至是一小部分。只要需求分析弄清楚了，概要设计详细设计完成了，那么编写相应程序是水到渠成，很自然的事。所以可能大家经常听说“软件蓝领工人”的说法，虽然有些夸张，但也表明编写程序在整个开发过程中已不是什么重要的技术活，似乎只是较为次要的体力活，只要需求、设计做好，编码是很简单的事。同样，以盖大楼为例，只要总设计师把方案定好，工程师把图纸绘好，民工们砌砖的事自然容易。

与“开发软件就是编写程序”类似的一个错误说法就是“软件就是程序”。规模较大的软件都是由多人分工协作开发出来的，为了使开发人员相互之间能准确地交换信息，同时也为了在软件交付给用户使用期间能比较容易地修改或扩充，必须把软件开发全过程中各个阶段的工作成果用文字、图表等形式准确地记录下来，这即是软件文档。所以软件应该由程序、数据和相关文档三部分共同组成。

错误观点二：“有一个对目标的概括描述就足以着手编写程序了，许多细节可以在以后再补充。”

持这种观点的人显然是轻视了需求分析的重要性。

事实上，对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的重要原因。只有用户才真正了解他们自己的需要，但是许多用户在开始时并不能准确具体地描述他们的需求，软件开发人员需要做大量深入细致的调查研究工作，反复多次地和用户交流信息，才能真正全面、准确、具体地了解用户的需求。对问题和目标还没有正确认识就仓促上阵，就如同写作文还没有弄清楚题意就匆忙下笔，即使写得再漂亮也是一篇不合格的跑题文章。软件开发工作同样遵循这个道理。这也就是为什么我们有时越早动手写程序，完成它所需要的时间反而越长的原因。

整个软件开发的过程可能并不是一帆风顺，有时会不可避免地发生一些变化，这些变化可能来源于用户方面的新需求，也可能来源于软件开发人员对之前用户需求理解的更正。不管怎样，出现了变化，就意味着要对之前的软件开发工作做出相应的修改，这本身也很正常。但问题的关键就在于变化出现的时机不同，修改所要付出的代价也是不同的。变化出现的时机越早，如在需求分析阶段出现的变化涉及改动的面较少，因而代价也比较低。反之，变化出现时机越晚，如在程序已经测试完毕才发觉所要完成的某个功能弄错了，这时返工的代价就相当高了。如图 1-4 表明了这种关系。其实，这就好比盖一栋大楼，在刚打地基的时候发觉需要修改户型结构和等到楼房已经盖到十层八层了才发觉需要修改户型结构，哪个时期所要付出

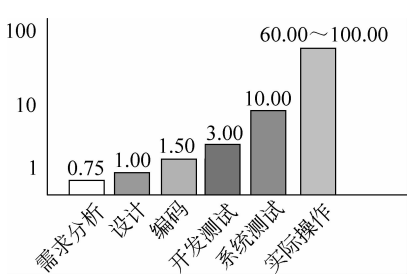

图 1-4　改正一个错误在不同阶段的代价