第 1 章　软件工程概述

1.1　软件危机

软件危机（Software Crisis） 是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

1.1.1　软件危机案例

案例一：1963 年，由美国宇航局发射的飞往火星的火箭爆炸，造成 1000 万美元的损失。原因是在一行代码中，将“ I=1，3 ”误写成“ I=1.3 ”。

案例二：1967 年，苏联“ 联盟一号 ”载人飞船在返航途中，在进入大气层时因打不开降落伞而烧毁。原因是软件忽略一个小数点导致。

案例三：1963－1966 年，IBM 公司开发了 OS/360 系统。该系统共有 4 000 多个模块，约 100 万条指令，投入 5 000 人/年，耗资数亿美元，结果还是延期交付。在交付使用后的系统中仍发现大量（2 000 个以上）的错误。原因是软件缺少适当的文档资料。

案例四：美国丹佛新国际机场自动化行李系统软件投资 1.93 亿美元开发，当时计划于 1993 年万圣节投入使用。但开发人员一直为系统错误困扰，屡次推后使用时间，直到 1994 年 6 月，机场计划者承认无法预测何时能启用。原因是软件开发进度难以预测。

案例五：1996 年，欧洲阿里亚纳 5 型运载火箭坠毁，造成 5 亿美元损失。原因是控制软件中出现了一个错误。

1.1.2　软件危机产生原因

有如下一些原因导致软件危机产生：

（1）对于软件开发的成本和进度的估计不准确。软件与硬件不同，软件是计算机系统的逻辑部件。由于缺乏软件开发的经验和软件开发数据的积累，使得开发软件的计划很难制订。主观盲目制订的计划，执行起来和实际情况会有很大的差距，使得开发经费一再增高；由于对工作量和开发难度估计不足，进度计划无法按时完成，开发时间也会一再拖延。同时客观上使得软件较难维护。

（2）软件开发是由多人分工合作并分阶段完成的过程，参与人员之间的沟通和配合十分重要。开发的软件产品不能完全满足用户需求，用户对已完成的软件系统不满意的现象常常

发生。导致这些现象的原因是软件开发人员在开发初期对用户的需求了解不够明确，未得到明确表达就开始着手编程。

（3）开发的软件可靠性差。开发和管理人员只重视开发而轻视对问题模型的定义，使软件产品无法满足用户的需求。由于在开发过程中，没有确保软件质量的体系和措施，在软件测试时，又没有经过严格的、充分的、完全的测试，使得提交给用户的软件产品质量差，在运行中暴露出大量的问题。这种不可靠的软件，轻则会影响系统正常工作，重则会发生事故，造成生命财产的重大损失。

（4）软件管理技术不能满足现代软件开发的需要，没有统一的软件质量管理规范。首先，文档缺乏一致性和完整性，从而失去管理的依据。因为程序只是完整软件产品的一个组成部分，一个软件产品必须由一组配置组成，不能只重视程序，而应当特别重视软件配置。其次，由于成本估计不准确，资金分配混乱，人员组织不合理，进度安排无序，导致软件技术无法实施。最后，软件的可维护性差。由于开发过程中没有统一的、公认的规范，软件开发人员按各自的风格工作，各行其是。因此，很多程序中的错误非常难改，实际上既不可能使这些程序适应新的硬件环境，也不可能根据用户需求在程序中增加新功能。

（5）软件开发提高的生产速度，远远跟不上计算机应用普及深入的趋势。软件产品“供不应求”的现象，使人类不能充分利用计算机硬件资源所提供的巨大潜力。

解决软件危机的途径有以下几种：

（1）加强软件开发过程的管理，构建良好的组织、严密的管理和协调工作的机制。

（2）推广使用开发软件的成功技术与方法，探索更好的、更有效的技术和方法，尽快消除在计算机系统早期发展阶段形成的错误概念。

（3）开发和使用好的软件工具，在适当的软件工具的支持下，开发人员可以更好地完成工作。

1.2 软件工程

解决软件危机既有技术措施又有管理措施，为了研究、解决软件危机，诞生了一门学科——软件工程学。它把软件作为工程对象，从技术措施和组织管理两个方面来研究、解决软件危机。1968 年，北大西洋公约组织的计算机科学家在德国召开国际会议，讨论软件危机问题，在这次会议上正式提出并使用了“软件工程”这个名词，一门新的工程学科就此诞生了。

软件工程（Software Engineering）是指导计算机软件开发和维护的一门工程学科，采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，经济地开发出高质量的软件并有效地进行维护。

软件工程的目标就是把软件作为一种物理的工业产品来开发，要求采用工程化的原理与方法对软件进行计划、开发和维护，摆脱不规范生产软件的状况，逐步实现软件开发和维护的自动化，开发出满足用户需求、及时交付、不超过预算和无故障的软件。

自从软件工程概念提出以来，经过几十年的研究与实践，虽然“软件危机”没得到彻底解决，但在软件开发方法和技术方面已经取得了很大的进步。尤其应该指出的是，20 世纪 80 年代中期，美国软件行业开始认识到，在软件开发中，最关键的问题是软件开发组织不能很好地定义和管理其软件过程，从而使一些好的开发方法和技术起不到所期望的作用。也就是说，在没有很好地定义和管理软件过程的软件开发中，开发组织不可能在好的软件方法和工具中获益。

1.3 软件生命周期

软件生命周期（Software Life Cycle，SLC）：一个软件产品从定义、开发、维护到废弃的时间总和称为软件的生命周期。

1.3.1 软件生命周期划分

软件生命周期是软件从产生到报废或停止使用的时间总和，包含问题定义、可行性研究、需求分析、概要设计（总体设计）、详细设计、编码、测试、维护等 8 个阶段。软件生命周期阶段的划分受软件的规模、性质、种类、开发方法等影响，阶段划分过细，会增加阶段之间联系的复杂性和软件开发、测试的工作量，在实际软件工程项目中较难操作。也有提出将软件生命周期划分成 4 个活动时期：软件分析时期、软件设计时期、编码与测试时期以及软件运行与维护时期。它们的关系如图 1-1 所示。

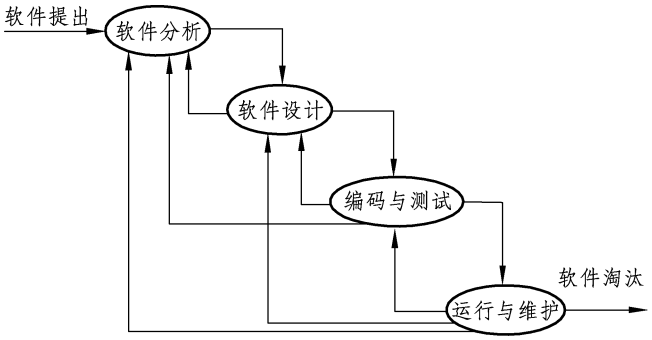

图 1-1 软件的 4 个活动时期

软件活动时期划分主要有如下几个优点：

（1）每个软件活动时期的独立性较强，任务明确且联系简单，容易分工。

（2）软件工程过程清晰、简明。

（3）软件规模大小都合适，大型软件可以在软件活动时期内再划分阶段进行。

（4）适合各种软件工程开发模型和开发方法。

（5）适合各类软件工程。

1.3.2 软件分析时期

软件分析时期也称为软件定义时期。这个时期的根本任务是确定软件项目的目标、软件应具备的功能和性能，构造软件的逻辑模型，并制定验收标准。在此期间，要进行可行性论证，并做出成本估计和经费预算，制定进度安排。通俗地说，软件分析时期主要解决如下问题：

（1）要做的是什么软件？

（2）有没有可行性？

（3）软件的具体需求是什么？

（4）验收标准是什么？

这个时期包括问题定义、可行性研究和需求分析三个阶段，可以根据软件系统的大小和类型决定是否细分阶段。

1. 问题定义

问题定义阶段要回答的问题是：要解决的问题是什么？如果不知道问题是什么就试图解决这个问题，最终得出的结果很可能是毫无意义的。

通过问题定义阶段的工作，系统分析员应该提出关于问题性质、工程目标和规模的书面报告。通过对系统的实际用户和使用部门负责人的访问调查，分析员扼要地写出对问题的理解，并在用户和使用部门负责人的会议上认真讨论这份书面报告，澄清含糊不清的地方，改正理解不正确的地方，最后得出一份双方都满意的文档。

2. 可行性研究

可行性研究阶段要回答的问题是对于上一阶段所确定的问题有行得通的解决办法吗？为此，系统分析员要进行压缩和简化的需求分析和设计，也就是在高层次上进行分析和设计，探索这个问题是否值得去解决，是否有可行的解决办法，最后要提交可行性研究报告。

经过可行性研究后制订项目开发计划。根据开发项目的目标、功能、性能及规模，估计项目需要的资源，即需要的计算机硬件资源、软件开发工具和应用软件包，需要的开发人员数目及行业背景要求，还要对软件开发费用做出估算，对开发进度做出估计，制订完成开发任务的实施计划。最后，将项目开发计划和可行性研究报告一起提交管理部门审查。

3. 需求分析

需求分析阶段的任务是准确地确定软件系统必须做什么，确定软件系统必须具备哪些功能。

用户了解他们所面对的问题，知道必须做什么，但是通常不能完整、准确地表达出来，也不知道怎样用计算机解决他们的问题。而软件开发人员虽然知道怎样用软件完成人们提出的各种功能要求，但是对用户的具体业务和需求却不清楚，这是需求分析阶段的困难所在。

系统分析员要和用户密切配合，充分理解用户的业务流程，完整、全面地收集、分析用户业务中的信息，从中分析出用户需求的功能和性能，完整、准确地表达出来。这一阶段要给出软件需求说明书。

软件分析时期结束前要经过管理评审和技术评审，合格后方能进入到软件设计时期。

1.3.3 软件设计时期

软件设计时期的根本任务是将分析时期得出的逻辑模型设计成具体计算机软件方案。具体来说，主要包括以下几个方面：

（1）设计软件的总体结构。

（2）设计软件具体模块的实现算法。

（3）软件设计评审。

软件设计结束之前要进行评审，评审通过后才能进入编码时期。理想的软件设计结果应该可以交给任何熟悉所要求语言环境的程序员编码实现。

软件设计时期也可以根据具体软件的规模、类型等决定是否细分成概要设计（总体设计）和详细设计两个阶段。

1. 概要设计

在概要设计阶段，开发人员要把确定的各项功能需求转换成需要的体系结构，在该体系结构中，每个成分都是意义明确的模块，即每个模块都和某些功能需求相对应。因此，概要设计就是设计软件的结构，在该阶段须明确以下几个问题：该结构由哪些模块组成？这些模块的层次结构是怎样的？这些模块的调用关系是怎样的？每个模块的功能是什么？同时还要设计该项目应用系统的总体数据结构和数据库结构，即应用系统要存储什么数据，这些数据是什么样的结构，它们之间有什么关系等。

这个阶段要考虑如下几种可能的方案：

（1）最低成本方案。系统完成最必需的工作。

（2）中等成本方案。不仅能够完成必需的任务，而且还有部分扩展功能。

（3）高成本方案。系统具有用户需要的所有功能。

系统分析员要使用系统流程图和其他工具描述每种可能的系统方案，用结构化原理设计合理的系统层次结构和软件结构。另外，系统分析员要估计每一种方案的成本与效益，在各种权衡的基础上向用户推荐一个最优的系统。

2. 详细设计

详细设计阶段就是为每个模块完成的功能进行具体描述，要把功能描述转变为精确的、结构化的过程描述。即该模块的控制结构是怎样的，先做什么，后做什么，有什么样的条件判定，有些什么重复处理等，并用相应的表示工具把这些控制结构表示出来。

1.3.4 编码与测试时期

编码与测试时期也称为软件实现时期。在这个时期，主要是组织程序员将设计的软件“翻

译”成计算机可以正确运行的程序，并且要按照软件分析中提出的需求和验收标准进行严格的测试和审查，审查通过后才可以交付使用。这个时期也可以根据具体软件的特点，决定是否划分成更细的一些阶段，如编码、单元测试、集成测试、系统测试、验收测试等。

1. 编码

编码阶段就是把每个模块的控制结构转换成计算机可接受的程序代码，即写成以某特定程序设计语言表示的“源程序清单”。写出的程序应是结构好、清晰易读，并且与设计相一致。

2. 测试

测试是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件的各个组成部分。测试分为单元测试、集成测试、系统测试和验收测试。

（1）单元测试又称为模块测试，目的是保证每个模块作为一个单元能正确运行，在这个测试步骤中所发现的往往是编码和详细设计的错误。通常由程序员自己来完成，用于查找各模块在功能和结构上存在的问题。

（2）集成测试，是在单元测试的基础上，将所有模块按照设计要求组装成为子系统或系统进行检查，主要是查找各模块之间接口上存在的问题。

（3）系统测试是在集成测试后进行，目的是充分运行系统，验证系统是否能正常工作并完成设计要求。

（4）验收测试又称为确认测试，是根据产品规格说明书严格检查产品，逐行逐字地按照说明书上对软件产品所做出的各方面要求进行，确保所开发的软件产品符合用户的各项要求。

1.3.5 运行与维护时期

软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求，如运行中发现了软件隐含的错误而需要修改，也可能是为了适应变化了的软件工作环境而需要做适当变更，还可能是因为用户业务发生变化而需要扩充和增强软件的功能等，要延续软件的使用寿命，就必须对软件进行维护。软件维护活动类型有四种：改正性维护、适应性维护、完善性维护、预防性维护。

改正性维护是指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误，改正性维护工作量占整个维护工作量的 17%～21%。

适应性维护是指使软件适应信息技术变化和管理需求变化而进行的修改。由于计算机硬件价格的不断下降，各类系统软件层出不穷，人们常常为改善系统硬件环境和运行环境而产生系统更新换代的需求，企业的外部市场环境和管理需求的不断变化也使得各级管理人员不断提出新的信息需求，这些因素都将导致适应性维护工作的产生。适应性维护工作量占整个维护工作量的 18%～25%。

完善性维护是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征，这些功能对完善系统功能非常必要。

完善性维护工作量占整个维护工作量的 50%～60%。

预防性维护是为了改进应用软件的可靠性和可维护性，为了适应未来的软硬件环境的变化，主动增加预防性的新的功能，以使应用系统适应各类变化而不被淘汰。预防性维护工作量占整个维护工作量的 4%左右。

1.4 软件工程思维培养

在项目开发中，如何运用软件工程的思维和方法思考问题是开发人员应具备的一种能力，从以下几方面进行考虑：

1. 考虑整个项目或者产品的市场前景

作为一个真正的系统分析人员，不仅要从技术的角度来考虑问题，而且还要从市场的角度去考虑问题。也就是说需要考虑产品的用户群范围，当产品投放到市场上的时候，是否具有生命力等。

2. 从用户的角度来考虑问题

从用户的角度考虑，用户认可的才是好的产品，并不是开发人员觉得好的就好。比如，一些对于开发人员来讲是显而易见的操作，但是对于普通用户来说可能就非常难以掌握，因此，需要在灵活性和易用性方面进行折中。比如尽管一些功能十分强大，但是如果用户几乎不怎么使用的话，就不一定在产品的版本发布时推出。

3. 从技术的角度考虑问题

技术是非常重要的，是成功的必要环节。在产品设计的时候，必须考虑采用先进的技术和先进的体系结构。比如，如果可以采用多线程进行程序中各个部分并行处理的话，就最好采用多线程处理；在 Windows 环境下开发的时候，能够把功能封装成一个单独的 COM 构件就不做成一个简单的 DLL 或者是以源代码存在的函数库或者是对象；能够在 B/S 结构下运行并且不影响系统功能的话就不用在 C/S 结构下实现。

4. 合理进行模块的分割

从多层模型角度来讲，一般系统可以分成用户层、业务层和数据库层三部分，当然每个部分都还可以再进行细分。在系统实现设计的时候，尽量进行各个部分的分割并建立各个部分之间进行交互的标准；在实际开发的时候，确实有需要的话再进行重新调整，这样就可以保证各个部分的开发齐头并进，开发人员也可以分工明确，各司其职。

5. 人员的组织和调度

软件开发中很重要的一点是要考虑人员的特长，有的人擅长图形用户接口开发，有的人喜欢做内核，要根据人员的具体情况进行具体配置。同时要保证每一个开发人员在开发的时候首先完成需要和其他人员进行交互的接口部分，并且对自己的项目进度以及其他开发人员的进度有一个清晰的了解，保证不同岗位的开发人员能够经常进行交流。

6. 开发过程中的记录

在开发过程中会碰到各种各样的困难，以及各种各样的创意，应该把这些东西记录下来并及时进行整理，对于困难和问题，如果不能短时间解决，可以考虑采用其他的技术替代，并在事后做专门的研究；对于各种创意，可以根据进度计划安排考虑是在本版本中实现还是在下一版本中实现。

7. 充分考虑实施时可能遇到的问题

开发好的软件产品是一方面，用户真正能够使用好产品又是另外一方面。比如在 MIS 系统开发中，最简单的一个问题就是如果用户输入数据错误，应如何进行操作。在以流程方式工作的时候，如何让用户理解自己在流程中的位置和作用，如何让用户真正利用计算机进行协作也是成败的关键。