第3章 软件工程过程模式

软件工程过程是软件产品开发的时间轨迹，沿着这个时间轨迹，基于项目的工程任务可以不断展开，并不断深入，与工程有关的各种工程方法、工具、标准、规程等诸多工程要素则能基于这个时间轨迹，逐渐结合到工程项目中来。既然是基于工程模式实施软件开发，其过程就必然需要有一定的时序结构，因此也就需要制定过程模式，以使得有关软件的工程开发的路线、步骤能够易于被所有的项目参与者理解与遵循。

本章要点：

- 软件生命周期。
- 瀑布模式、原型进化模式、增量模式。
- 螺旋模式、迭代模式、组件复用模式。

3.1 软件生存周期

生命的一般过程是胚胎、萌芽、生长、成熟，然后逐渐衰老并走向消亡，体现在软件上则是规划、定义、开发、运行、维护，直到最终失去应用价值而退役。

软件生存周期是对软件生命过程的工程标准说明，可作为软件开发、运行、维护的基本的工程化过程框架。软件生存周期的国际标准是 ISO/IEC 12207—1995，基于该国际标准而建立的最新国家标准是 GB/T 8566—2007。

GB/T 8566—2007 从多个不同立场对软件生存周期进行了细节说明，涉及产品需求方的行为过程、产品供应方的行为过程、产品开发方的行为过程、产品运作方的行为过程、产品维护方的行为过程。

通常情况下，基于软件产品开发方的立场，软件生存周期可体现为 3 个生命段：定义期、开发期、运行与维护期。

3.1.1 软件定义期

软件定义期是软件项目初期时段，涉及确立软件项目、制订项目计划、分析用户需求等几项任务，下面是对这几项任务的说明。

1. 确立软件项目

软件产业开发涉及供需双方。能够提供产品的是供应方，需要获得产品的则是需求方。当产品开发是根据需求方委托进行定制时，产品供需双方大多是通过招投标方式形成产品供需合同，由此确立产品规格，并以此为依据进行产品开发。

委托定制软件产品的一般立项过程是：

(1) 软件需求方编制软件产品项目招标书,以明确表达对软件产品的应用需求;

(2) 软件供应方研究需求方项目招标书,并确定承接该项目的可行性,如果确认项目具备可行性,则编制软件产品投标书,以响应需求方的产品招标书;

(3) 如果软件供应方投标书能够被需求方接受,则双方可签订产品合同,以明确供需双方责权关系。

当软件产品开发是基于一般市场需求进行研制时,也需要进行软件项目立项。不同于委托定制的是,这时的需求方通常是开发机构的较高决策层,而供应方则是开发机构的较低开发层,相互之间不对等,并不能建立对等合同,而是通过产品需求报告、产品开发立项通知、产品开发任务通知等形式确立软件项目。

通用产品研制的一般立项过程是:

(1) 开发机构中的市场研究部门,根据市场需求情况编制产品需求报告;

(2) 开发机构较高决策层审核产品需求报告,如确认则进行立项可行性研究;

(3) 如果该产品经研究确认具备开发可行性,则发文确立开发项目,并通知对口开发部门或项目组承接该项目任务。

立项过程中最重要的工作是进行项目可行性分析,以对软件项目可否实施做出可行性评估。开发软件是有风险的,项目可行性分析有利于事先确知风险的高低。可行性分析时需要建立软件系统高层分析模型,其用作可行性分析对象,可行性分析内容则有技术可行性、经济可行性和应用可行性。

2. 制订项目计划

一旦软件供应方承接了软件需求方的软件开发任务,接下来的首要工作就是制订项目计划,以使软件项目后续工作能够有序开展。

软件项目计划是对项目的全局性规划,需要基于项目合同或任务书编制。

软件项目管理机构需要审核确认项目计划,并以此为依据监督项目实施。

3. 分析用户需求

分析用户需求,就是搞清楚用户对软件有什么要求,以使得开发出来的软件能够满足用户的实际需要。

这项工作原则上应由软件需求方完成,然而,如果是完全的产品委托开发,则作为最终用户的软件需求方,一般只会提供有关用户需求的框架说明,因此,更多的需求细节还需要软件供应方给予完善。

通常情况下,软件开发者可按照以下步骤分析用户需求。

(1) 研究用户需求框架;

(2) 调查用户,获取用户需求细节;

(3) 完善用户需求,建立需求模型,涉及功能模型、数据模型;

(4) 基于软件需求模型进行全面的软件规格定义。

(5) 编制“软件需求规格说明书”,作为今后软件开发与验证的依据。

3.1.2 软件开发期

软件开发者的核心任务是制造软件产品。当通过需求分析确定软件规格之后,就要按规格要求开发软件,包括软件设计、编码、测试,直到最终交付软件产品。

1. 软件概要设计

软件设计的第一步是概要设计。这是针对软件系统的结构设计，用于从总体上对软件的构造、接口、全局数据等给出设计说明。

程序模块是构造软件的基本元素。概要设计中的软件结构即建立于程序模块基础上。

实际上，不同的设计方法、软件系统有不同的模块元素。在结构化设计中，程序函数、过程等是基本模块元素。在面向对象设计中，程序类、对象等就是基本模块元素。概要设计时并不需要搞清楚这些程序模块的内部细节，但模块的功能、数据特征以及模块之间的调用和引用关系则需要在概要设计中确定下来。

概要设计需要提交“概要设计说明书”作为概要设计任务标志性成果，也是后续详细设计与系统集成的依据。

2. 软件详细设计

软件设计的第二步是详细设计，以概要设计为依据，需要解决的问题是程序模块内部的执行流程与数据构造。

设计出优良的、高效率的程序算法是详细设计的核心任务。

算法设计者可通过专门的算法描述工具（如流程图、PAD 图、伪码等）说明程序算法，设计结果则通过提交“详细设计说明书”，作为程序编码的依据。

3. 编码和单元测试

在软件详细设计完成之后，接着需要按照详细设计说明书的要求进行程序编码。

如果详细设计说明书中程序的算法描述详细周全，则程序编码就成了一件比较简单的语言转译工作，甚至可以利用软件工具自动生成程序代码。

程序编码的同时还需要进行程序调试。

针对单元模块的单元程序测试往往和编码结合在一起进行，需要以详细设计说明书为依据，对单元模块在功能、算法与数据结构上是否符合设计要求进行确认。

4. 系统集成

系统集成也称为系统组装。系统集成的任务是按照概要设计中的软件结构，并基于某种集成策略（如渐增集成策略），将诸多创建出的程序模块装配成一个系统。

在系统集成过程中，还需要对系统进行集成测试，以确保系统结构符合设计要求。

5. 系统验收

系统验收是用户对系统的确认验证。系统验收将以需求规格说明书为依据，需要验证软件产品与规格定义的一致性，包括第一步的 Alpha 测试与第二步的 Beta 测试。

Alpha 测试是在开发环境下进行的系统验证，测试过程由开发者操控，但需要用户参与。

Beta 测试则是在用户实际应用环境中由用户独立进行的系统验证，开发者将不做操控。

3.1.3 软件运行与维护期

开发出来的软件产品在验收通过之后就可以交付用户使用了。对于已交用户使用的软件，开发机构仍需要承担一定的后期维护，以确保软件系统的正常运行。

软件系统的维护主要有：

(1) 改正性维护。改正性维护是对程序中出现的错误的修正。

(2) 适应性维护。适应性维护是对运行环境的改变而做的适应性修改。

(3) 完善性维护。完善性维护是对已建系统的功能补充或完善。

3.2 瀑布模式

瀑布模式是传统的软件开发过程模式,其中“瀑布”形象地表达了该模式自顶向下、逐级细化的过程特征。

瀑布模式任务流程如图 3-1 所示,立项论证是软件项目起步时的顶,通常是对软件问题初步的模糊认识,后续阶段可对模糊软件问题逐步求解,直至获得可运行软件产品。

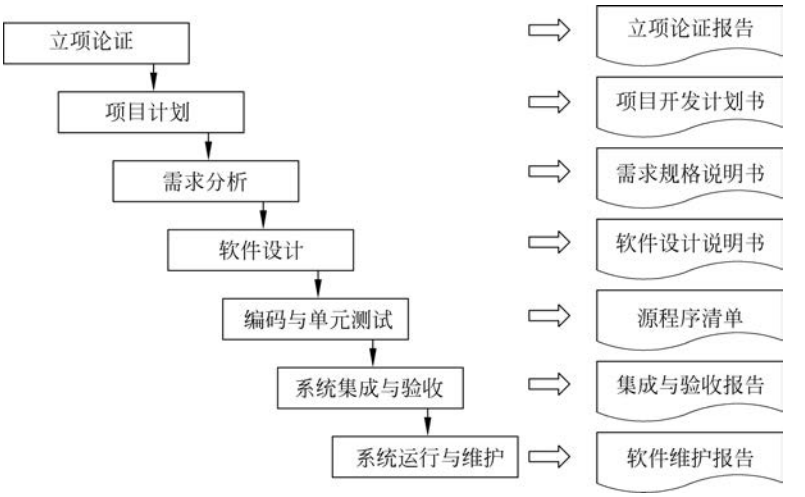

图 3-1 瀑布模式任务流程

3.2.1 瀑布模式的特点

1. 线性化过程

瀑布模式开发过程是一个无支路的线性化过程,涉及分析、设计、编码、集成等几个阶段。瀑布模式要求各阶段任务之间严格按衔接次序逐级推进,不允许跨越阶段任务,必须等到上一阶段任务完成之后,下一阶段任务才能开始。

2. 里程碑管理

瀑布模式中的每个阶段都有确定的与任务相关联的成果,如分析阶段的分析报告、设计阶段的设计说明书、编程阶段的源程序,它们可作为各阶段里程碑标志,用于标识阶段任务的结束。

基于里程碑的管理可带来项目进程量化。实际上,项目管理者可事先估算出每个阶段的任务量,并以此为依据规定每个阶段的任务时限,由此而达到对项目进程的量化管理。

3. 阶段评审

瀑布模式中的各阶段成果都需要进行严格的质量评审,以确保每个阶段都能达到预期的任务目标。

4. 文档驱动

瀑布模式中前一阶段产生的软件文档将成为后一阶段的工作基础与约束条件。这也就是说，可依靠文档使项目由前一阶段推进到后一阶段。

3.2.2 瀑布模式中的信息反馈

图 3-1 所示的瀑布模式是理想化的过程模式，其建立在各阶段任务无差错的基础上，要求每个阶段都能达到预期目标，都能取得预期结果，因此无须回头再做以前阶段的工作，而只需考虑项目往前推进。然而，实际项目则并不可能绝对完美，用户的需求变更、开发者自身的工作遗漏等都会使项目不得不返回到以前阶段。

显然，为了方便重做以前阶段的工作，后期对前期的问题发现需要能够反馈到前期工作中去。因此，过程模式中需要有合适的信息反馈通道。

图 3-2 所示即为带有信息反馈通道的瀑布模式，当前一阶段的软件问题在后续阶段被发现时，能够沿着信息反馈通道逐级返回，并在问题解决之后再将修正结果逐级下传。

图 3-2 带有信息反馈通道的瀑布模式

应该说，增加的信息反馈通道使理想化的线性瀑布模式有了非线性化的改变，但其非线性化过程只是局部的，仅限于相互衔接阶段之间。实际上，为了确保产品的可追踪性，任何变更行为都被限制于相邻阶段之间逐级进行。例如，在编程实现阶段发现了需求遗漏，则该问题需先返回给设计师，然后通过设计师再返回给分析师。若该问题有了新的需求解决方案，则新的方案需先通知设计师，然后再由设计师通知到编程人员。

3.2.3 瀑布模式的作用

1. 应用价值

瀑布模式是获得广泛应用的软件开发过程模式，具有简洁的便于工程应用的线性化过程步骤，并可通过里程碑管理机制使项目进程量化。

瀑布模式的作用还体现在基于里程碑而建立起来的质量保证体系上。每个阶段结束前都要对所完成的阶段成果进行评审，这使得软件错误能够在各阶段内尽早被发现和解决。在涉及工程目标的诸多因素中，质量总是第一因素，因此，具有良好质量保证机制的瀑布模型也就有了很强的生命力。

瀑布模式是经典过程模型，为其他过程模式的产生提供了一个较好的模式标本，如 3.4

节将要介绍的增量模式就吸收了瀑布模式的优点。

2. 局限性

瀑布模式是线性化过程,只能按规程推进,并且必须等到所有开发任务完成以后才能获得可以交付使用的软件产品。因此,瀑布模式并不能快速创建软件系统,对于一些急于交付的软件系统,瀑布模式有操作上的不便。

瀑布模式可较好地适应需求明确的软件系统开发,但瀑布模式的灵活性不是很好,如果已经开始设计,则来自于用户的一个很小的需求变更请求就可能会给软件项目带来大难题,导致项目延期。实际上,大多数的应用系统在开发初期,用户的需求并不清晰,因此,对于那些面向用户的应用系统的开发,瀑布模式有较大的不适应性。

3.3 原型进化模式

3.3.1 软件原型

软件原型就是对软件问题的直观模拟或仿真。软件分析或设计时都可能用到软件原型。根据生命周期的长短,软件原型可分为抛弃型原型和进化型原型。

1. 抛弃型原型

抛弃型原型大多用来获取需求评价或对设计做试探,之后就失去了使用价值。因此,抛弃型原型一般使用软件工具快速生成,以降低制作成本。

由于抛弃型原型是一个无须投入实际应用的实验品,因此可针对某个专门问题建立局部原型,而无须考虑其完整性,如针对人机交互建立界面原型、针对业务步骤建立工作流原型、针对数据汇总建立 SQL 查询原型。

2. 进化型原型

进化型原型是可改进并能最终演变为可交付产品的原型,一般要求在最终产品开发平台上创建。

开发者通常选择可视化开发工具(如 Microsoft Visual Basic、Visual Studio . NET、C++ Builder、JBuilder 和 Dreamweaver)创建进化原型,原因是这些可视化开发工具不仅能快速创建软件原型,而且还能使软件原型投入实际应用,并逐步演变为最终目标系统。

界面原型是用得最多的进化型原型。通常情况下,开发者先建立界面原型供用户做应用操作评价,并在获得用户的原型确认后,接着设计和实现与该应用操作有关的内部功能执行模块。

3.3.2 原型进化过程

进化型原型可贯穿整个开发过程,因此也就可考虑与其相适应的过程模式,以达到更有效的工程应用。

一种合理的过程是,开发者建立可供用户使用的原型系统,然后收集用户对原型的使用评价,并以此为依据逐步对原型系统进行修正,逐步使其接近并最终达到目标系统的要求。

1. 原型进化的特点

图 3-3 所示是原型进化开发过程,由该图可发现原型进化过程具有以下一些特点。

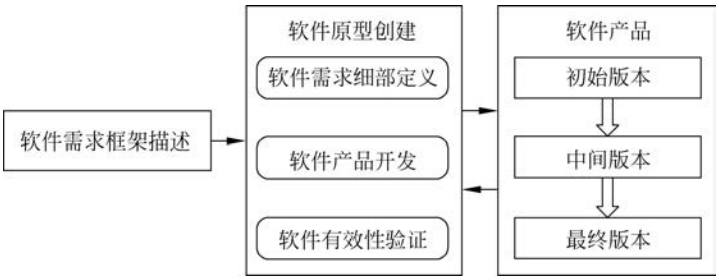

图 3-3　原型进化开发过程

(1) 在对软件问题做初步分析并获得有关软件的需求框架之后,即可进行原型创建。其中的需求框架一般来自用户需求的收集与整理,它是直接面向用户的,如业务组织、业务域、业务流程、基本功能点等。

(2) 软件开发就是原型创建,其包括需求细化、产品开发和产品验证等多项任务。这些任务是在同一个工作进程内并行或交替进行的。实际上,软件的分析者、设计者、编码者处在同一个工作空间,并且一个人可扮演分析、设计、编码等多重角色。用户也可加入原型创建中来,因此,开发者与用户之间可以有非常便利的意见交流。

(3) 通过发布新版本而使原型系统逐步修正与完善。实际上,一个并不完整的初始版本即可投入使用,以方便软件急用之需,然后通过版本更新而逐步地满足用户对于软件的多方面需要。

2. 原型进化的缺陷

(1) 不能建立里程碑管理,以致项目进度难以量化,并使软件质量难以得到有效控制。虽然可通过原型初始版本使软件尽早投入使用,但什么时候能够获得可满足全面需求的最终版本则难以确定下来。

(2) 虽然可通过发布新版而适应用户需求变更,但版本的快速更替也使软件配置管理变得复杂起来。版本的快速变更还可能会给软件结构带来损伤,使软件结构缺乏整体性。

(3) 对于面向用户的中小型软件开发,原型进化模式有一定优势。然而也有管理规程上的不足,并不能有效保证软件质量,因此不能很好地适应大型软件系统的开发。

3.4　增 量 模 式

3.4.1　增量开发过程

图 3-4 所示是增量开发过程,分为设计结构、开发增量构件、集成系统 3 个任务域。

(1) 设计结构:对系统进行全局分析与设计,通常以增量构件为单位定义需求框架,并以此为依据确定软件系统体系结构。

(2) 开发增量构件:对软件体系结构中的增量构件进行需求细化,并以此为依据实现与验证增量构件。

(3) 集成系统:基于软件体系结构进行构件集成与系统级验证。重复第(2)步,继续下一个构件的开发与集成,直到完成全部构件的创建与集成。

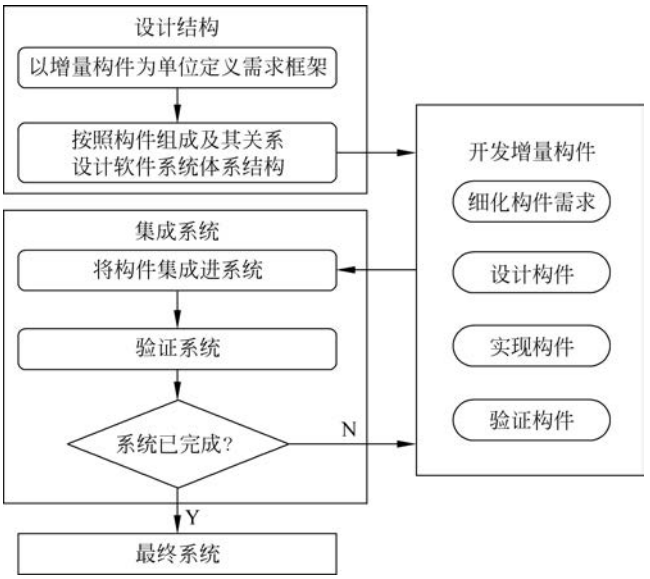

图 3-4　增量开发过程

3.4.2　增量模式的优越性

3.2 节和 3.3 节介绍了瀑布模式与原型进化模式。瀑布模式的优点是有较完善的工程管理机制,可有效保证软件质量,不足则是较难应付用户需求变更,并必须等到项目任务全部完成之后才能交付软件产品;原型进化模式则刚好相反,优点是可适应用户需求变更,并可较快创建应用系统,不足是缺乏有效的里程碑流程管理,不能适应大型系统开发。

显然,更有效的过程模式应具有两者的长处,既有较完善的工程管理机制,又能适应用户需求变更。增量模式即有这样的优越性,整体上具有里程碑管理特性,有利于质量监控;局部则基于构件构造,有利于逐步构建与完善,并可适应用户需求变更。

更具体地看,增量模式体现出了以下优越性。

(1) 项目前期工作容易开展,仅依靠需求框架,如业务域、业务流程、基本功能点等,即可设计系统构架。

(2) 基于任务域实现里程碑流程控制,能较好地保证软件质量,并可适应大型应用软件系统的开发。

(3) 直到开发构件时才需考虑需求细节,有利于用户需求的逐步明朗,并对构件级需求变更有较好的适应。

(4) 可按照构件的功能价值安排开发顺序,并逐个实现与交付。因此,一些用户急需的功能可优先开发,并尽早投入应用。

(5) 系统基于构件逐渐扩充,利于开发者经验的逐步积累,并利于技术有效复用。实际上,已使用的算法、技术策略、源码等都可更有成效地应用到构件创建中去。

(6) 可利于降低项目的技术风险。通常,最先交付的最具优先权的核心构件会受到最多次数的集成测试,并带来核心构件最高的可靠性。

3.5 螺旋模式

螺旋模式是一种可较好规避开发风险的过程模式。

软件开发有来自各个方面的风险，如用户需求风险、技术经验风险、产品质量风险等。显然，如果能够很好地识别风险，并能事先制定应对风险的措施，则风险的危害性就会显著降低。

螺旋模式的特点是项目基于任务域螺旋式递进。图 3-5 所示即为螺旋模式，其中的螺旋线用来表示项目进程，每一个螺旋回路对应一个过程任务域，从内至外分别是需求分析、软件设计、系统集成、验证与交付。

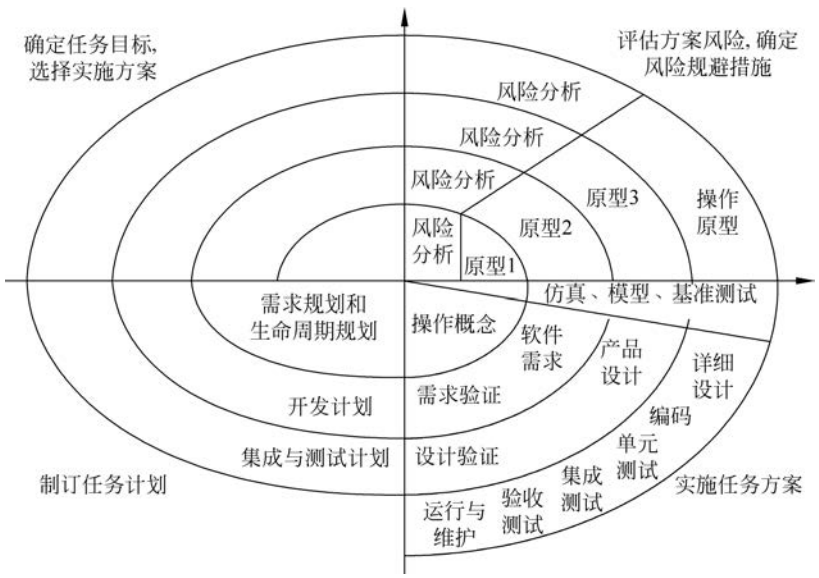

图 3-5　螺旋模式

螺旋模式中的每一个任务域都需要进行风险评估，并需要根据评估结论制定风险规避措施。

通常情况中，每个任务域涉及以下几个步骤。

（1）制订任务计划。

（2）确定任务目标，选择实施方案。

（3）评估方案风险，确定风险规避措施。

（4）实施任务方案。

值得考虑的是，对软件项目进行风险分析也是需要费用的，若软件项目风险分析费用过高，甚至超过了软件开发费用，则软件项目风险分析在经济上就不合算了。一般只有大规模并有较高风险的软件项目，才有采用螺旋模式的必要。例如，需要开发一个核反应堆控制程序，则这个程序就不能出现任何差错，否则后果不堪设想。因此，这个程序开发过程就必须有严格的对于技术质量的风险识别与应对措施。

3.6 迭代模式

迭代模式如图 3-6 所示。这是一种任务可叠交的过程模式。所谓迭代,也就是软件的分析、设计与实现可交替反复进行。因此,传统瀑布模式中的清晰任务边界在迭代模式中变得模糊起来。

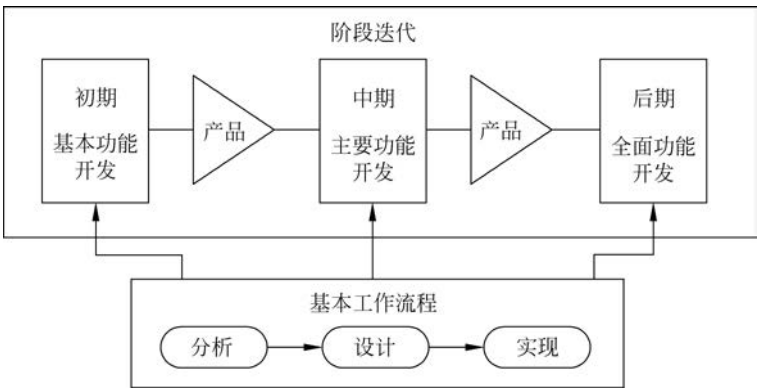

图 3-6 迭代模式

迭代模式将项目任务分为初期、中期、后期等多个阶段。通常,初期阶段只需要创建具有必备功能的软件系统;中期阶段则是在初期结果基础上做进一步完善,以获得更好的软件应用;后期阶段则是在中期结果基础上进行更全面的开发,并以实现软件全部功能为建设目标。显然,迭代模式更有利于兼顾近期目标与远期规划。初期阶段体现为实现近期目标,后期阶段则体现为实现远期规划。

一般看来,迭代模式对面向对象方法有更好的过程支持,可使面向对象方法获得更有成效的工程应用。

面向对象的技术特点是基于类体构造对象系统,并且软件分析或设计时都需要定义类体。因此,面向对象方法可基于类体使分析、设计顺畅交替。类体还可派生出下级子类体,并可依靠子类体进行系统扩充。

经验表明,面向对象方法基于类的分析与设计的交替,以及由类体派生子类而进行的系统扩充,为实现迭代开发提供了便利。

3.7 组件复用模式

所谓软件复用,就是利用已有软件要素构造新的软件系统。传统结构化方法采用的是功能函数复用。这是一种直接建立在程序代码基础上的小粒度复用技术,方法是建立能够广泛用于各种程序的标准函数。然而,大多数的标准函数只能提供一些简单功能,并且缺乏必要的柔性,因此适应范围受到限制。

面向对象方法采用的是类继承复用。一般看来,类继承复用有比函数复用更好的适应性,子类不仅可继承父类方法,而且可重新定义方法。但类继承复用受程序语言限制,子类必须与父类有相同的程序语言。

基于组件的集成则带来了组件复用，并反映出了更好的复用效果。

(1) 组件经过编译，能够跨语言应用；

(2) 组件是大粒度复用，更具工程复用价值；

(3) 显著提高了软件生产率。

实际上，组件复用带来了流水线软件装配，系统所需组件大多无须专门开发，可通过专业制作机构提供。例如，要开发一个商业数据汇总系统，其界面可由 A 公司提供组件，业务流程配置可由 B 公司提供组件，报表生成则由 C 公司提供组件，而项目组只需将这些组件集成起来，就可实现系统开发。

图 3-7 所示是基于组件复用的过程支持，其涉及以下 6 个步骤的工作任务。

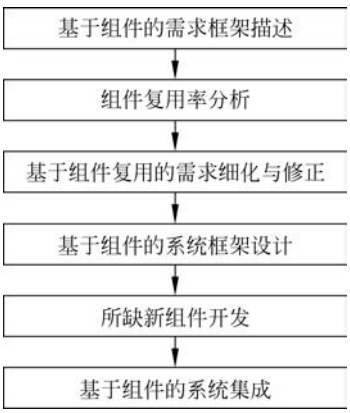

图 3-7　基于组件复用的过程支持

(1) 基于组件的需求框架描述：系统有哪些功能？系统可划分为哪几个组件域？系统由哪些组件构成？

(2) 组件复用率分析：已有哪些组件？需要购买哪些组件？需要开发哪些组件？

(3) 基于组件复用的需求细化与修正：通过对需求框架的细化而获得对软件系统的基于组件的详细需求定义。

(4) 基于组件的系统框架设计：确定基于组件的系统构架。

(5) 所缺新组件开发：开发那些需由项目组自己创建的新组件。

(6) 基于组件的系统集成：基于组件进行系统集成，以建立软件系统。

小　　结

1. 软件生命周期

软件生存周期是软件由提出到开发再到投入应用的全过程，基于开发者立场一般划分为 3 个生命段：定义期、开发期、运行与维护期，每个生命段又包含若干阶段任务。

2. 瀑布模式

瀑布模式是最传统的过程模式，“瀑布”形象地表达了自顶向下、逐级细化的过程特征。

瀑布模式的特点是：线性化过程；里程碑管理；阶段评审；文档驱动。对于需求明确的软件项目，瀑布模式有较好的适应性。然而，如果需求不明确或需求易变更，则瀑布模式就显现出了不适应性。

3. 原型进化模式

原型进化模式的开发流程是开发者先建立原型系统供用户评价或使用，然后根据用户的意见反馈对原型系统不断修正，由此使它逐步接近并最终达到目标系统的要求。

原型进化模式可较好地适应用户的需求变更，但却因缺乏里程碑管理机制而不能很好地支持大型项目。

4. 增量模式

增量模式是瀑布模式与原型进化模式优点的结合，其将系统分解为多个增量构件，然后

以增量构件为原型部件,逐步创建、集成与完善。

增量模式在整体上具有瀑布模式的里程碑特点,可适应大型项目。但在系统的局部构建上,则体现为基于增量构件的原型进化,可适应用户的需求变更。

5. 螺旋模式

螺旋模式是一种可较好地规避开发风险的过程模式。螺旋模式的特点是项目基于任务域螺旋式递进,每个任务域都需要进行风险评估,并需要根据评估结论制定有效的风险规避措施。

6. 迭代模式

迭代模式是软件的分析、设计与实现可交替反复进行的过程模式。迭代模式对面向对象方法有更好的过程支持,可使面向对象方法获得更有成效的工程应用。

7. 组件复用模式

组件复用模式是对基于组件的系统集成的过程支持。组件复用可带来流水线软件装配,系统所需组件大多无须专门开发,而可通过专业制作机构提供,由此可提高软件开发效率,并可提高软件产品质量。

习　　题

1. 软件开发期的目标任务是什么?概要设计需要完成什么任务?

2. 瀑布模型的一大特点是里程碑管理机制。对此,请谈谈你的认识。

3. 为什么瀑布模型不能很好地适应用户需求变更?原型进化模式为什么又能很好地适应用户需求变更?

4. 试说明抛弃型原型与进化型原型的异同。

5. 一般认为,原型进化模型不能适应较大型软件项目的开发,其原因是什么?

6. 增量模式结合了瀑布模型与原型进化模型的优点。请具体说出增量模式体现出哪些方面的优越性?

7. 螺旋模式的优越性是什么?采用螺旋模式的代价是什么?

8. 为什么迭代模式能够较好地适应面向对象开发?

9. 组件复用模式有什么特点?

10. 某大型企业计划开发一个“综合信息管理系统”,涉及销售、供应、财务、生产、人力资源等多个部门的信息管理。该企业的想法是按部门优先级别逐个实现,边应用边开发。对此,需要一种比较合适的过程模型。请对这个过程模型做出符合应用需要的选择,并说明选择理由。