# 第3章

# 软件生存周期模型

软件工程的核心就是过程，如图 3-1 所示。软件产品、人员、技术通过过程关联起来。软件工程过程能够将软件生存周期内涉及的各种要素集成在一起，从而使软件的开发能够以一种合理而有序的方式进行。

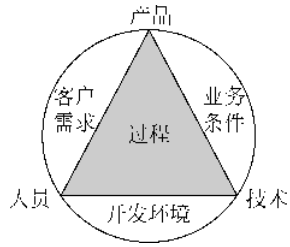


**图 3-1　软件工程过程的核心元素**

在 ISO/IEC 12207 标准中，规定了一个完整的软件生存命周期应该包括哪些过程，过程中应该包括哪些活动来保证质量。至于什么时候实施什么过程/活动，反复几次合适则根据项目特点定义。在具体实践中，开发者往往通过一些模型来刻画软件生存周期。在 ISO/IEC 12207 标准中，软件生存周期模型是指一个包括软件产品开发、运行和维护中有关过程、活动和任务的框架，其中这些过程、活动和任务覆盖了从该系统的需求定义到系统的使用终止。

把这个概念应用到软件开发过程中，可以发现所有的模型都具有如下基本特征：

- 描述了开发的主要阶段。
- 定义了每一个阶段要完成的主要过程和活动。
- 规范了每一个阶段的输入和输出(提交物)。
- 提供了一个框架，可以把必要的活动映射到该框架中。

根据过程模型提出的时间不同，可以把软件过程模型分为传统软件工程过程模型和现代软件工程过程模型。

传统软件工程过程模型的主要代表是编码修正模型、瀑布模型、增量模型、演化模型和螺旋模型；Rational 统一过程(RUP)、敏捷过程(AP)和微软解决方案(MSF)等则是现代软件工程过程模型的主要代表。下面重点介绍几种典型的软件开发生存周期模型。

根据软件模型所要求的规范的全面程度，又可把过程模型分为计划驱动的模型和敏捷模型，如瀑布模型、统一过程模型(以下简称 RUP)等为计划驱动的模型，极限编程(简称 XP)为敏捷模型。本章首先介绍最原始的模型——编码修正模型，然后介绍瀑布模型、增量模型、演化模型、螺旋模型、RUP 模型 6 个计划驱动的模型，然后介绍在这些模型中，被广泛

采用的原型构造、并行开发和商业组件 3 种方法。

## 3.1 编码修正模型

编码修正模型是所有模型中最古老的也是最简单的模型，如图 3-2 所示。该模型将软件开发过程分成编码和测试两项活动。在编码之前几乎不做任何预先计划的工作，该模型的使用者很快就进入所开发产品的编码阶段。典型情况是，完成大量的编码后测试产品并且纠正所发现的错误。编码和测试工作一直持续到产品开发工作全部完成并将产品交付给客户。

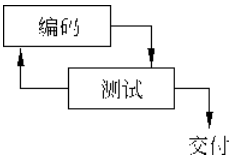


**图 3-2 编码修正模型**

该模型的主要特点是：

- 最适用于很小且简单的项目。编码修正模型是从一个大致的想法开始工作，然后经过非正规的设计、编码、调试和测试方法，最后完成工作。
- 成本可能很低，经过少量设计就进入编码阶段。
- 易于使用，人员只需要很少的专业知识，任何写过程序的人都可以用。
- 对于一些非常小的、开发完后就会很快丢弃的软件可以采用。
- 对于规模稍大的项目，采用这种模型是很危险的。由于缺乏预先的计划并且通常伴随着不正规的开发方式，容易导致代码碎片，交付的产品质量也很难保证。且因为设计没有很好文档化，因此代码维护困难。

## 3.2 瀑布模型

瀑布模型(Waterfall Model)是典型的软硬件开发模型如图 3-3 所示。它包括需求、设计、编码、测试、运行与维护几个阶段。在每一阶段提交以下产品：软件需求规格说明书、系统设计说明书、实际代码和测试用例、最终产品、产品升级等。工作产品流经“正向”开发的基本步骤路径。“反向”的步骤流表示对前一个可提交产品的重复变更。由于所有开发活动的非确定性，因此是否需要重复变更，这仅在下一个阶段或更后的阶段才能认识到。这种“返工”不仅在以前阶段的某一地方需要，而且对当前正在进行的工作也同样需要。

该模型的主要特点是：

- 每一阶段都以验证/确认活动作为结束，其目的是尽可能多地消除本阶段产品中存在的问题。
- 在随后阶段里，尽可能对前面阶段的产品进行迭代。

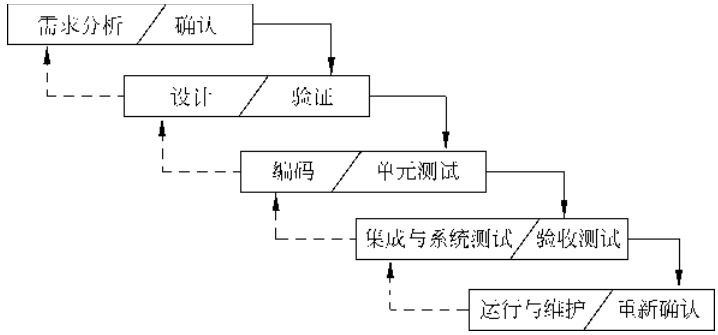


图 3-3　瀑布模型

### 3.2.1　瀑布模型的优缺点

瀑布模型是第一个被完整描述的过程模型，是其他过程模型的鼻祖。其优点是：

- 容易理解、管理成本低。瀑布模型的主要成果是通过文档从一个阶段传递到下一个阶段，各阶段间原则上不连续也不交叠，因此可以预先制定计划，来降低计划管理的成本。
- 它不提供有形的软件成果，除非到生存周期结束时。但文档产生并提供了贯穿生存周期的进展过程的充分说明。允许基线和配置早期接受控制。前一步作为下一步被认可的、文档化的基线。

其不足是：

- 客户必须能够完整、正确和清晰地表达其需要。但在系统开发中经常发现用户与开发人员沟通存在巨大差异、用户提出含糊需求但又被开发人员随意解释，以及用户需求会随着时间推移不断变化等问题。
- 可能要花费更多的时间来建立一些用处不大的文档。
- 在开始的两个或三个阶段中，很难评估真正的进度状态。
- 在一个项目的早期阶段，过分强调了基线和里程碑处的文档。
- 开发人员一开始就必须理解其应用。
- 当项目接近结束时，出现了大量的集成和测试工作。
- 直到项目结束之前，都不能演示系统的能力。

瀑布模型是传统过程模型的典型代表，因为管理简单，常被获取方作为合同上的模型。一个阶段完成后，生产出一个具体的产品；如果需要的话，可以对这一产品进行独立的检验。获取方组织可以按每一阶段向开发方组织支付费用，这意味着双方必须客观地对其完成情况进行核实。

当一个项目有稳定的产品定义和很容易被理解的技术解决方案时，可以使用瀑布模型。

在这种情况下，瀑布模型可以帮助你及早发现问题，降低项目的阶段成本。它提供开发者渴望的稳定需求。若要对一个定义得很好的版本进行维护或将一个产品移植到一个新的平台上，那么瀑布模型是快速开发的一个恰当选择。

对于那些容易理解但很复杂的项目，采用纯瀑布模型比较合适，因为这样可以用顺序的方法处理复杂的问题。在质量需求高于成本需求和进度需求的时候，瀑布模型表现得尤为出色。由于在项目进展过程中基本不会产生需求的变更，因此，纯瀑布模型避免了一个常见的、巨大的潜在错误源。

### 3.2.2 V模型

瀑布模型的一个变体就是V模型（如图3-4所示），它在每一个环节中都强调了测试（并提供了测试的依据），同时又在每一个环节都做到了对实现者和测试者的分离。由于测试者相对于实现者的关系是监督、考察和评审，因此测试者相当于在不断地进行回顾和确认。

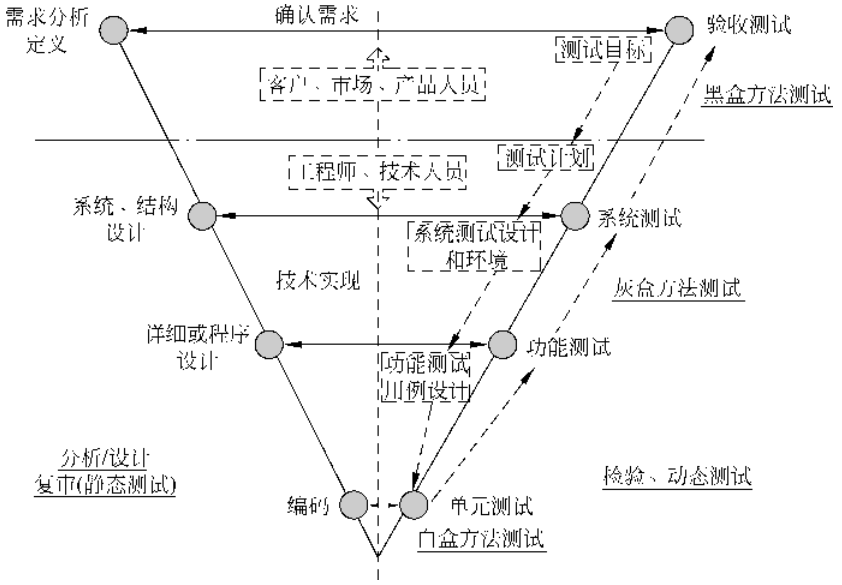


**图3-4 V模型示意图**

在图3-4中，左半部分是设计和分析，是软件设计实现的过程，同时伴随着质量保证活动——审核的过程，也就是静态的测试过程；图3-4的右半部分是对左边结果的验证，是动态测试的过程，即对设计和分析的结果进行测试，以确认是否满足用户的需求。如：

- 需求分析和功能设计对应验收测试，说明在做需求分析、产品功能设计的同时，测试人员就可以阅读、审查需求分析的结果，从而了解产品的设计特性、用户的真正需求，确定测试目标，可以准备用例并策划测试活动。

- 当系统设计人员在做系统设计时，测试人员可以了解系统是如何实现的以及基于什么样的平台，这样就可以设计系统的测试方案和测试计划，并事先准备系统的测试环境，包括硬件和第三方软件的采购。因为这些准备工作实际上是要花去很多时间的。
- 当设计人员在做详细设计时，测试人员可以参与设计，对设计进行评审，找出设计的缺陷，同时设计功能、新特性等各方面的测试用例，完善测试计划，并基于这些测试用例来开发测试脚本。
- 在编程的同时进行单元测试是一种很有效的办法，这样做可以尽快找出程序中的错误，充分的单元测试可以大幅度提高程序质量，降低成本。

从图 3-4 可以看出，V 模型中的质量保证活动和项目同时展开，项目一启动，软件测试的工作也就启动了，从而避免了瀑布模型在代码完成之后才进行软件测试的弊端。

- 图 3-4 的水平点划线上部表明，其需求分析、定义和验收测试等主要工作是面向用户的，要和用户进行充分的沟通和交流，或者是和用户一起完成。相对来说，水平点划线下部的大部分工作都是技术性工作，在开发组织内部进行，主要由工程师、技术人员完成。
- 从垂直方向看，越在下面，白盒测试方法使用越多，集成与系统测试大多将白盒测试方法和黑盒测试方法结合起来使用，形成灰盒测试方法。在验收测试过程中，由于用户一般要参与，所以使用黑盒测试方法。

V 模型被广泛应用于软件外包中。由于劳动力短缺等多种原因，很多企业把项目直接外包给国内/国外的开发团队，项目成果的阶段性考察则成为其第一要务，因为这直接决定了何时、如何以及由谁来进入下一个环节。

因此 V 模型变得比其他模型更为实用。模型的左端是接受外包任务的团队或者公司，而右端则是外包的软件企业中有丰富经验的工程人员。这样既节省人力，又可以保证工程质量。事实上，即使图 3-4 左端的外包任务是由多个团队同时承接，右边的工程人员也不需要更多的投入。

## 3.3　增量模型

增量生存周期模型(Incremental Life Cycle Model)是第一个由瀑布模型演变而来的，它是对瀑布模型的精化。该模型有一个假设，即需求可以分段，成为一系列增量产品，对每一增量可以分别地开发，如图 3-5 所示。

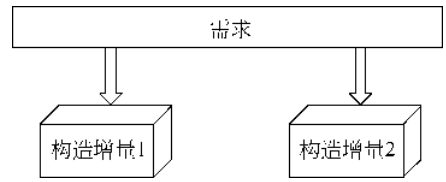


**图 3-5　增量模型**

在开始开发时，需求就很明确，并且产品还可

以被适当地分解为一些独立的、可交付的软件，称为构造增量；在开发中，期望尽快提交其中的一些增量产品。在一些情况下，通常采用增量模型。如一个数据库系统要求通过不同的用户界面，为不同类型的用户提供不同的功能。在这一情况下，首先实现完整的数据库设计，并把一组具有高优先级的用户功能和界面作为一个增量；以后，陆续构造其他类型用户所需求的增量。

图 3-6 表达了如何利用瀑布模型来开发增量模型中的构造增量。尽管该图表示对不同增量的设计和实现完全可以是并发的，但在实际应用中，可以按任一期望的并行程度进行增量开发。如可以在完成了第一个增量设计之后，吸取经验教训，再转向第二个增量的设计。图 3-7 给出了一个应用实例。

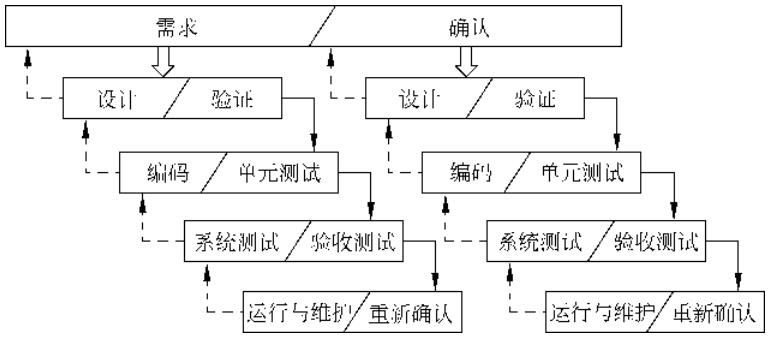


**图 3-6　增量的瀑布开发模型**

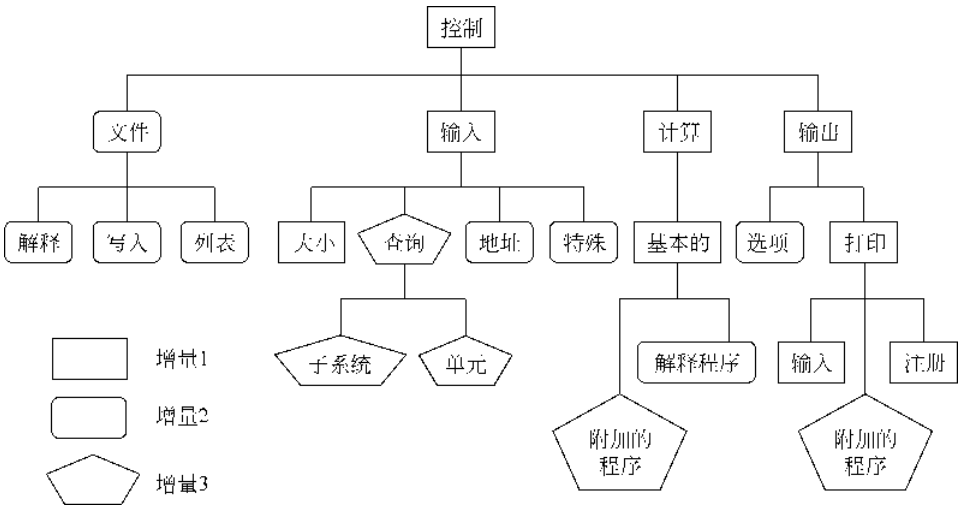


**图 3-7　增量模型应用实例**

如果一个增量并不需要交付给客户的话，那么这样的增量通常称为一个“构造”。如果增量需要被交付，那么它们就被认为是发布版本。在编写软件生存周期计划时，不论是正式

的还是非正式的，都要注意使用客户期望的术语，其表达要与合同和工作陈述保持一致。

增量模型作为瀑布模型的一个变体，具有瀑布模型的所有优点，此外，它还有以下优点：

- 第一个可交付版本所需要的成本和时间很少。
- 开发由增量表示的小系统所承担的风险不大。
- 由于很快发布了第一个版本，因此可以减少用户需求的变更。
- 允许增量投资，即在项目开始时，可以仅对一个或两个增量投资。

然而，如果增量模型不适于某些项目，或使用有误，则有以下缺点：

- 如果没有对用户的变更要求进行规划，那么产生的初始增量可能会造成后来增量的不稳定。
- 如果需求不像早期考虑到的那样稳定和完整，那么一些增量就可能需要重新开发、重新发布。
- 管理发生的成本、进度和配置的复杂性，可能会超出组织的能力。

从以上第一点和第三点可以看出，如果客户的变更要求与以前的增量相矛盾，双方容易发生冲突。因此在采用此模型时，开发方需要有合适的配置管理和成本计算系统，并在合同中明确给出“变更条款”。如果出现问题，可以依据合同进行处理，化解矛盾。

如果采用增量投资方式，那么客户就可以对一些增量进行招标。然后，开发人员按提出的截止期限进行增量开发，因此，可以用多个契约来管理组织的资源和成本。

当需要以增量方式开发一个具有已知需求和定义的产品时，可以使用增量模型。优点是产品的各模块在很大程度上可以彼此并行开发，从而可以在开发周期内尽早地证明操作代码的正确性而降低产品的技术风险。注意在项目中并行执行的活动数量提高，则管理项目的复杂度就会加大。

## 3.4　演化模型

演化模型是显式地把增量模型扩展到需求阶段。从图 3-8 可以看出，为了第二个构造增量，使用了第一个构造增量来精化需求。这一精化可以有多个来源和路径。

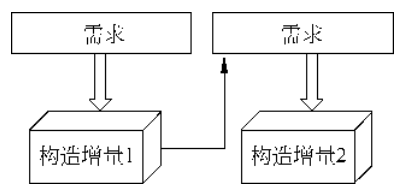


**图 3-8　演化生存周期模型**

首先，如果一个早期的增量已向用户发布，那么用户会以变更要求的方式提出反馈，以支持以后增量的需求开发。第二，通过实实在在地开发一个构造增量，为以前还没有认识到的问题提供了可见性，以便实际开始这一增量工作。

在演化模型中，仍然可以使用瀑布模型来管理每个演化增量。一旦理解了需求，就可以像实现瀑布模型那样开始设计阶段和编码阶段。

使用演化模型不能够成为弱化需求分析的借口。在项目开始时，应考虑所有需求来源的重要性和风险，对这些来源的可用性进行评估。只有采用这一方法，才能识别和界定不确定需求，并识别第一个增量中所包含的需求。此外，合同条款应该反映所采用的开发模型。例如，对每一个增量的开发和交付，双方应该按照合同进行协商，包括下一个增量的人力成本和费用的选择。

同样，成本计算、进度控制、状态跟踪和配置管理活动必须能够支持这一模型。由于演化的增量具有明确的顺序，因此与增量模型相比，演化模型面临的挑战通常是较弱的。但应认识到，一定程度的并发总是存在的，因此系统必须允许某一层次的并行开发。

演化模型的长处和不足与增量模型类似。演化模型还具有以下优点：

- 在需求不能予以规范时，可以使用这一演化模型。
- 用户可以通过运行系统的实践，对需求进行改进。
- 与瀑布模型相比，需要更多用户/获取方的参与。

演化模型的缺点有：

- 演化模型的使用仍然处于初步探索阶段，因此具有较大的风险，需要有效的管理。
- 该方法的使用很容易成为不编写需求或设计文档的借口，即使需求或设计可以得到很清晰的描述。
- 用户/获取方不理解该方法的自然属性，因此当结果不够理想时，可能产生抱怨。

当需求和产品定义没有被很好地理解，以及需要更快地开发和创建一个能展示产品外貌和功能的最初版本时，特别适合使用演化模型，这些早期的增量能帮助用户确认和调整需求及帮助他们寻找相应的产品定义。

演化模型与增量模型具有许多相同的优点，而且还具有能适应产品需求变更的显著优点，它还引进了附加的过程复杂性和潜在的更长的产品周期。

## 3.5 螺旋模型

螺旋模型（如图 3-9 所示）是由 Boehm 提出的另一种生存周期过程模型。在这一模型中，开发工作是迭代进行的，即只要完成了开发的一个迭代过程，另一个迭代过程就开始了。

该模型关注解决问题的基本步骤，由此可以标识问题，标识一些可选方案，选择一个最佳方案，遵循动作步骤，并实施后续工作。尽管螺旋模型和一些迭代模型在框架和全局体系结构上是等同的，但它们所关注的阶段及活动是不同的。

开发人员和客户使用螺旋模型可以完成如下工作：

- 确定目标、方案和约束。
- 识别风险和效益的可选路线，选择最优方案。
- 开发本次迭代可供交付的内容。

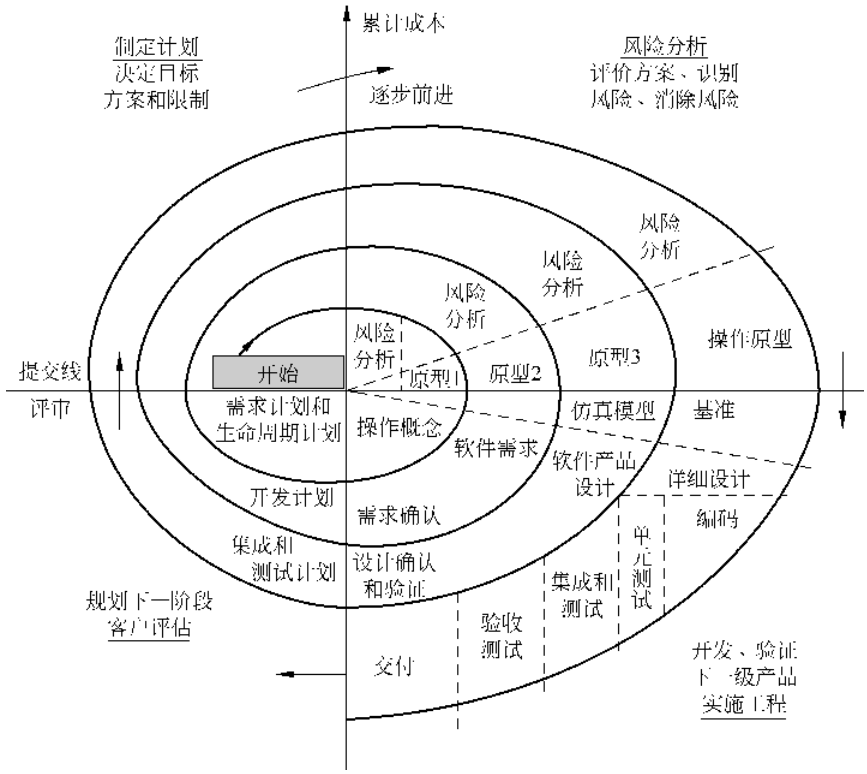


**图 3-9　螺旋模型**

- 评估完成情况，规划下一个迭代过程。
- 交付给下一步，开始新的迭代过程。

螺旋模型扩展了增量模型的管理任务范围，因为增量模型基于以下假定：需求是最基本的，并且是唯一的风险。在螺旋模型中，决策和降低风险的空间是相当广泛的。

螺旋模型的另外一个特征是：实际上只有一个迭代过程真正开发可交付的软件。如果项目的开发风险很大，或客户不能确定系统需求时，螺旋模型就是一个好的生存周期模型。

螺旋模型强调了原型构造。需要注意的是，螺旋模型不必要求原型构造，但原型比较适合于这一过程模型。

在螺旋模型中，把瀑布模型作为一个嵌入的过程。即分析、设计、编码和交付的瀑布过程，是螺旋一周的组成部分。

螺旋模型是一种以风险为导向的生存周期模型，它把一个软件项目分解成一个个小项目。每个小项目都标识一个或多个主要风险因素，直到所有主要风险因素都被确认。“风险”的概念在这里是有外延的，它可以是需求或者构架没有被理解清楚、潜在的性能问题、根本性的技术问题，如此等等。在所有的主要风险因素被确定后，螺旋模型就像瀑布模型一样

终止。

在螺旋模型中，越早期的迭代过程成本越低。规划概念比需求分析的代价低，需求分析比开发设计、集成和测试的代价低。

在螺旋模型中，项目范围逐渐增量展开。项目范围展开的前提是风险被降低到仅仅是下一步扩展部分的、可以接受的水平。在该模型中，要进行几次迭代以及每次迭代中通常采用几个步骤完成并不重要，尽管那是很好的工作次序。重要的是要根据项目的实际需求调整螺旋的每次迭代过程。

可以采取几种不同的方法把螺旋模型和其他生命周期模型结合在一起使用，通过一系列降低风险的迭代过程来开始项目。在风险降低到一个可以接受的水平后，可以采用瀑布模型或其他非基于风险分析的模型来推断开发效果。可以在螺旋模型中把其他过程模型作为迭代过程引入。如当遇到“不能确定性能指标是否能够达到”的风险时，可以使用原型法来验证是否能达到目标。

螺旋模型最重要的优势是随着成本的增加，风险程度随之降低。时间和资金花得越多，风险越少，这恰好是在快速开发项目中所需要的。

螺旋模型提供至少和瀑布模型一样多或更多的管理控制。该模型在每个迭代过程结束前都设置了检查点。模型是风险导向的，对于无法逾越的风险是可以预知的。如果项目因为技术和其他原因无法完成，可以及早发现，这并不会使成本增加太多。

螺旋模型比较复杂，需要责任心、专注度和管理方面的知识，通过确定目标和可以验证的里程碑，来决定是否启动下一轮开发。在有些项目中，产品开发的目标明确、风险适度，就没有必要采用螺旋模型提供的适应性和风险管理。

## 3.6 原型构造在生存周期模型中的应用

在生存周期模型的发展中，下一步自然是：显式地规划如何使用一个或多个演化增量，作为一个明确的需求开发工具，这样一个增量称为一个原型。尽管原型可以由用户以某一受限的方式使用，但不能把它看成是一个具有完备功能的增量。

在软件开发中，构造原型的目的有二：其一是开发需求，具备完备功能的、可交付的、可支持的增量的实现就是基于这些需求，这些需求包括功能需求和界面需求；其二是用于为一个项目或一个项目的某些部分确定技术、成本和进度可行性。如一个原型有助于回答：

- 一个新的开发环境或工具，是否能够满足客户成本和进度约束？
- 一个被安装的、可用的软/硬件基础设施，是否可以支持客户新的性能和能力需求？
- 是否能够创建这一产品，即这是可行的吗？

原型构造在那些具有用户界面和数据库的系统中被普遍使用。

## 3.7　生存周期模型中并发的作用

当一个项目经历其选择的生存周期时，一些过程之间的重叠几乎是不可避免的，不论这一重叠是规划的还是没有规划的。如需求可能是非常清楚的，出于对成本和进度的考虑，可能选择了通常的瀑布模型。从表面上看，这排除了并发的可能性，但事实上这是非常困难的。如一个子系统的详细设计在另一个子系统之前完成，这是很常见的事。在这种情况下，如果两个子系统之间所提供的接口是稳定的，那么提前进入已完成详细设计的那个子系统的编码是很合情理的事情；并且，完成详细设计的开发人员经常负责同一个子系统的编码，而不是去帮助完成第二个子系统的详细设计；从而导致了一个系统的详细设计阶段和编码阶段的并发。而人们往往认为这一并发过程是顺序的。

组织管理系统必须有能力支持并发，该系统包括进度安排、成本控制、状态跟踪和配置管理，包括技术复审机制和任何设计工具。当两个以上子系统同时处于一样的开发阶段时必须严格监控其接口定义。

另外，在所有生存周期模型的反向流中，还可能隐含地存在一些并发。这是客观存在的。围绕并发，需要考虑并发程度和并发管理两方面问题。

## 3.8　商业组件和复用的作用

现代软件系统的创建趋势是：使用商业应用框架和商业组件，或复用组织内部已开发的组件和框架。当然，也可以复用组织的实践和规程。这一趋势的出现，有以下 3 个原因：

- 市场和成本的竞争压力。
- 交付环境的日趋复杂和标准化。
- 产品线工程的出现，其中，应系统地规划和执行多个相关软件产品的开发和演化，在产品线的所有成员中复用部分设计和实现。

这种趋势对一个项目生存周期过程具有或大或小的影响。如若开发组织使用一个新的商业应用框架来构建一个产品，那么就需要开发一个原型，以获取框架使用的经验，并检验该框架对这一应用的适应性。如果使用的框架是组织内部开发的，那么也需要开发一个原型，以评估该框架对应用开发的适应性。框架的选择就是这样一个基本的设计决策，以至于必须在项目生存周期的早期实施原型构造。如果可能的话，原型构造应在承约项目的成本和进度之前实施。

如果使用内部开发的组件或从市场购买的组件作为新系统的组成部分，那么就必须在项目早期评估这些组件的适应性。确切地说，就是怎样使用这些组件以及什么时候评估这