第3章　软件生命周期模型

软件产品通常由一些模糊的概念引出的，例如，“要是计算机能画出曲线图就好了”，“如果公司对我们每天的现金流动没有明确的认识，那么我们在6个月以后将无法偿还债务”，或者是“如果我们开发这种新型的电子表格并投放市场，我们将会赚100万美元”。一旦产品的需求建立之后，产品将进入一系列的开发阶段。典型地，产品将经历规格说明、设计和实现过程。如果客户对产品满意，产品将被安装。而在运行过程中，产品还要经历维护。当产品最终达到有效寿命的终点时，就会退役。产品所经历的这一系列步骤，称为“生命周期模型”(life-cycle model)。

每个产品的生命周期的历史是不同的。有些产品会在概念阶段渡过几年的时间，这也许是因为硬件速度不够快，无法运行该产品，也许是因为在开发一个高效的算法之前，必须进行一些基础研究。有些产品的设计和实现很快，而维护时间可能达许多年，以便随用户的需要不断做修改。但还有些产品，在经过设计、实现和多年维护后，由于维护的花费很大，使开发一个全新的产品比试图修补当前版本更便宜。

本章将描述许多不同的软件周期生命模型。使用最广的两种模型是瀑布模型和快速原型模型。此外，螺旋模型目前也正受到越来越多的注意。为有助于评价这三种模型的优劣，本章还介绍了其他的生命周期模型，包括增量模型和令人极不满意的边做边改(build-and-fix)模型。

3.1　边做边改模型

很遗憾，许多产品都是使用称为“边做边改”模型开发的。这些产品没有规格说明，也没有经过设计，而且随着客户的需要一次又一次地不断修改。这种方法如图3-1所示。虽然边做边改模型对于100行或200行的短程序来说还不错，但这种方法对任何规模合理的产品来说则完全不能令人满意。如图1-3所示，在需求规格说明或设计阶段修改产品，其费用相对较小，但如果在产品已编写好代码后，或更坏地，在产品已处于运行状态时，再修改产品，则其费用将高得难以承受。因此，边做边改的方法所用经费远远大于经过正确规格说明和设计的产品费用。此外，没有规格说明文档或设计文档，产品的维护将极其困难，产生回归故障的机会也将大大增加。

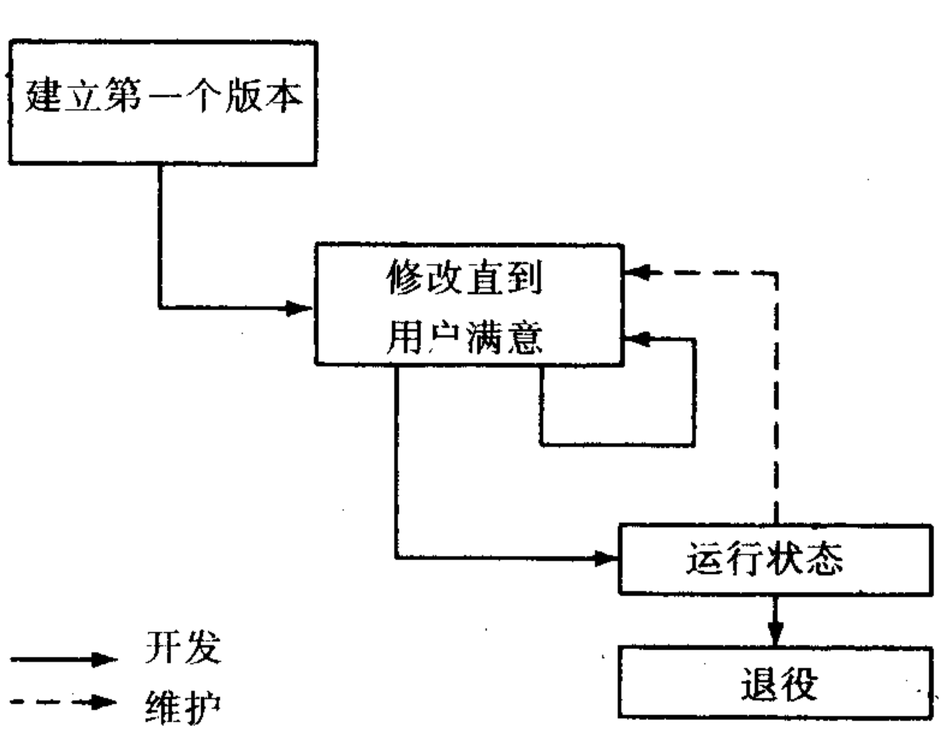

图3-1　边做边改模型

若不使用边做边改方法，就必须在产品开发之前选定总体的行动计划，即选定生命周期模型。生命周期模型(有时简称“模型”)确定软件过程的各种不同的阶段(如需求、规格说明、计划、设计、实现、集成和维护等阶段)及各个阶段的执行顺序。一旦生命周期模型获得所有成员的同意，就可以开始开发产品了。

直到80年代早期，瀑布模型一直是唯一被广泛接受的生命周期模型。现在开始详细地讨论这种模型。

3.2　瀑布模型

瀑布模型(waterfall model)首先由Royce[Royce,1970]提出。该模型的形成如图3-2所示。首先是确定需求，并接受客户和SQA小组成员的验证。然后拟定产品的规格说明，即形成说明产品要做什么的文档。当文档由SQA小组验证并得到客户的认可后，这个阶段便告成功。一旦客户在规格说明文档上签字，计划阶段就可开始。该阶段的任务是草拟一份软件开发的详细时间表。所拟定的计划也必须在SQA小组检验。当客户同意了开发人员的期限估计和费用估计后，设计阶段就开始了。规格说明文档描述产品做什么，而设计文档则描述产品要怎么做。

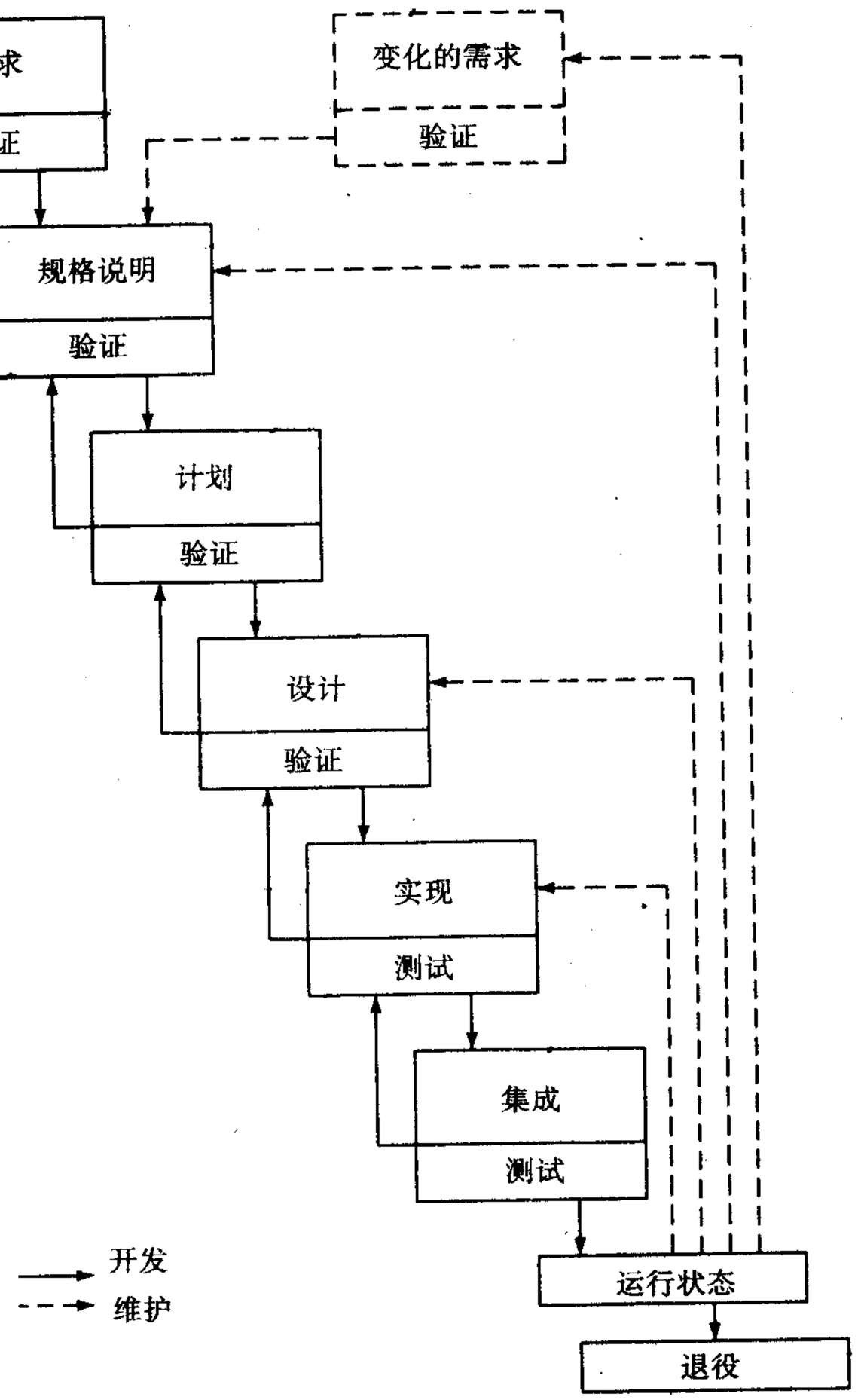

图3-2　瀑布模型

在设计阶段，有时能明显地看出规格说明文档中的错误。规格说明可能是不完备的(产品的特征被忽略)、自相矛盾的(规格文档中两条或多条声明对产品的定义互不相容)或含糊不清(规格说明文档有一种以上的解释)。由于不完备、自相矛盾、含糊不清的存在，所以有必要在继续开发前校订规格说明文档。再参照一下图3-2所示，从设计框左边往回指向计划框，再往回指向规格说明框的箭头，构成了一个反馈循环。若开发人员必须在设计阶段修正规格说明文档，则可遵循这一循环。于是在客户的允许下，就可对规格说明文档做必要的修改，同时调整计划和设计文档以融入这些修改。在设计人员最终觉得满意后，就把设计文档交给程序员去实现。

设计上的缺陷可能会在实现过程中显示出来。例如，一个实时系统的设计可能在实现后发现速度太慢。比如，这样的设计缺陷在FORTRAN中可能是由于数组b中的元素是按列序存储的，即以b(1,1),b(2,1),b(3,1),...,b(n,1),b(1,2),b(2,2),b(3,2),...,b(n,2),...顺序存储的。假定200 × 200的FORTRAN数组b按一列存储在一块中的方式存储，即每执行一次FORTRAN read命令时，将把一列中的200个字读入内存缓冲区。假定要将整个数组从磁盘读入主存中。如果数组一列一列地读入，则要读入所有40 000个数组元素，就要读整整200个块。第一条read命令将把第一列放到缓冲区中，于是前面200条read命令将使用缓冲区中的内容。只有在要用到第201个元素时，才需要将第2块从磁盘传输到主存中。但是，如果产品按一行一行的顺序读取数组，则每条read命令都将读入一个新块，因为这时候每个read都访问不同的列，也就是访问不同的块。因此，读入整个数组要传输40 000个块，而当数组按列主序读入时，只需要传输200个块，这样，产品部分的输入/输出时间将增加200倍。这种类型的设计失误必须在进行下一步开发前得到修正。

在实现阶段，如果有必要，带反馈环的瀑布模型允许开发人员修改设计文档、软件项目管理计划，甚至规格说明文档。在实现期间，将对模块进行编码实现，制作文档，然后将各模块合并成一件完整的产品(在实践中，实现阶段和集成阶段通常是并行进行，每个模块被实现和测试后，就马上集成起来，见第13章)。在集成过程中，有时可能有必要返回来修改代码，也许在这之前还要修改规格说明和设计文档。

瀑布模型中至关重要的一点是，只有当一个阶段的文档已编制好，且该阶段的产品得到了SQA小组的认可时，该阶段才算完成。这一点也适用于修改活动，如果由于遵循反馈环而使前一阶段的产品必须做修改，则必须等该阶段的文档已做了修改，且本次修改得到了SQA小组的验证后，这个前期阶段才算完成。

当开发人员对完成的产品感到满意后，就将产品交给客户进行验收测试。在这个阶段交付的还包括用户手册及其他在合同中列出的文档。当客户确认该产品确实满足规格说明文档后，该产品将移交给客户，进行安装，进入运行状态。

一旦客户接受了产品，其后的任何修改，无论是排除残余故障，还是以某种方法扩展该产品，都属于维护。从图3-2中可以看出，维护可能不仅仅要求作实现上的改变，而且可能需要设计和规格说明上的改变。此外，需求的变化会触发增进维护，而增进维护又通过规格说明文档、设计文档和代码来实现。瀑布模型是一个动态模型，反馈环在这个动态机制中起着重要的作用。对文档必须像对代码本身一样，要细心地维护，在下一阶段开始之前，对前一阶段的产品必须仔细地检查。这同样也是至关重要的。

瀑布模型已在各类产品中获得了巨大的成功。然而，失误也是有的。要决定一个工程是否使用瀑布模型，必须先理解它的优势和不足。

瀑布模型分析

瀑布模型有许多优点，其中包括：近乎强迫式的规则方法，在每个阶段必须提供文档的约

束和要求每个阶段的所有产品(包括文档)必须由SQA小组仔细地验证。标志每个阶段结束的必不可缺少的一点是SQA小组对该阶段所有产品(包括文档)的认可。

测试是瀑布模型各个阶段所固有的。测试不是在产品制作完成后才执行的一个独立的阶段，也不是在每个阶段最后执行的，而是例如第2章所描述，测试应该在整个软件过程中持续地进行。特别是，需求在被确定后也必须验证，就像规格说明文档和软件项目管理计划一样。代码必须用各种方法来测试。在维护期间，不仅仅要保持产品修改后的版本仍具有原来版本的功能，而且应当仍可正确地执行(回归测试)，而且要保证它完全满足客户所加强的任何新的需求。

规格说明文档、设计文档、代码文档和其他文档(如数据库手册、用户手册和操作手册)是产品维护所必不可缺少的工具。如第1章所描述，软件预算中平均有67%用在维护上。遵守瀑布模型的文档约束，将使维护更容易。如上一节所强调的，维护期间也必须和软件生产时一样，遵守条理清晰的方法。每项改动都必须反映在相关的文档中。瀑布模型的成功很大程度上是由于它基本上是一种文档驱动的模型。

然而，瀑布模型是文档驱动的这事实也可能是一个缺点。要了解这一点，请看一下下面的两种情况。第一，Joe和Jane Johnson决定建造一所房子。他们向一位建筑师请教。建筑师没有给他们看房子的框架、计划和模型，而是给他们20页用极具技术性的术语描述这所房子的文档。尽管Joe和Jane都没有建筑方面的经验，而且几乎不了解文档的内容，但他们还是激动地签了字，说："好吧，就建造这座房子！"

另一个情形是，Mark Marberry通过邮购买衣服。服装公司没有给他寄公司衣服的图案和可选布料的样品，而是给Mark寄来了关于裁剪方面和产品所用布料书面的描述资料。其后，Mark仅仅根据这些书面的描述定购了一套衣服。

这两种情形似乎是根本不可能出现的。不过，许多采用瀑布模型开发的软件的确正是这样制作的。整个过程从规格说明开始。一般来说，规格说明文档是冗长的、详细的、非常直接的，读起来很烦琐。客户通常没有阅读软件规格说明方面的经验，而由于规格说明文档通常以一种客户所不熟悉的风格书写，这更是难上加难。如果规格说明是以Z[Spivey, 1992](见8.8节)之类形式化的规格说明语言书写的，那就更难办。不过，不管是不是正确理解了，客户都会在规格说明文档上签字了事。在许多方面，Joe和Jane Johnson同意按他们只能部分理解的书面描述来建造房子，和客户批准一个以他们只能部分理解的规格说明文档描述来开发软件产品，这两者之间并没有什么区别。

Mark Marberry和他邮购的衣服的事情似乎极为罕见，但是，这正是瀑布模型在软件开发中运用时所发生的事情。客户第一次看到的运行产品只是整个产品已编好代码之后所交付的产品。有点令人奇怪的是，软件开发人员害怕听到这样的话，"我知道这是我所要求做的，但不是真正想要的。"

究竟出了什么错呢？客户理解由规格说明文档描述的产品的方式，与他们理解实际产品的方式有很大的不同。规格说明只存在纸上，因此客户不能真正知道产品本身是什么样子。由于瀑布模型几乎完全依赖书面的规格说明，这会导致产品的制作可能不能满足客户的真正需要。

应该指出，正如建筑师可通过模型、框架和计划来帮助客户理解要建造什么，软件工程师也能利用图表技术来与客户交流，如数据流图(第8.3.1节)。问题是这些辅助图表不能描述最终的产品将如何工作。例如，流程图(产品的图表说明)和运行产品本身之间有很大的不同。

下面考察快速原型模型，其优势在于有助于确保客户的真实需要能得到满足。

3.3　快速原型模型

快速原型(rapid prototype)是一个运行模型，它在功能上等价于产品的一个子集。例如，如

果目标产品管理应付款、应收款和货物入库，则快速原型可能执行数据捕捉的屏幕处理和打印报告，但没有文件更新和出错控制。对于一个用于判断溶液中酶的浓度的目标产品，其快速原型可能是执行计算并显示结果，但不对输入数据进行有效性或合法性检查。

如图3-3所示，快速原型生命周期模型的第一步是建造一个快速原型，让客户和未来的用户与之交互，并试用它。一旦客户觉得这个快速原型确实做了大多数做的事情，开发人员便可以拟定规格说明文档，并担保该产品能满足客户的真实需要。

构建好快速原型之后，软件过程便按图3-3所示的顺序继续下去。快速原型模型的主要优势在于，产品的开发基本上是线性的，从快速原型到交付的产品，其间没有也不需要瀑布模型中的反馈环(图3.2所示)。这有许多原因。第一，开发组成员们根据快速原型来构造规格说明文档。由于运行中的快速原型已通过与用户交互得到验证，所以有理由说所产生的规格说明文档是正确的。第二，考虑设计阶段。即使快速原型装配得很匆忙(也很恰当)，设计组也能从中观察到许多东西，最坏情况下也至少能知道“如何不去做什么”。于是，这里几乎不需要反馈环。

下一阶段是实现。在瀑布模型中，设计的实现有时会暴露出设计错误。在快速原型中，由于建立了一个初步的运行模型，会减少在实现期间或之后修正设计的需要。原型将给设计组带来方便，即使它也许只反映了最终目标产品的部分功能。

一旦客户验收了产品并安装了之后，维护便开始了。按必须执行的维护种类不同，周期可能会返回到需求阶段、规格说明、设计阶段或实现阶段。

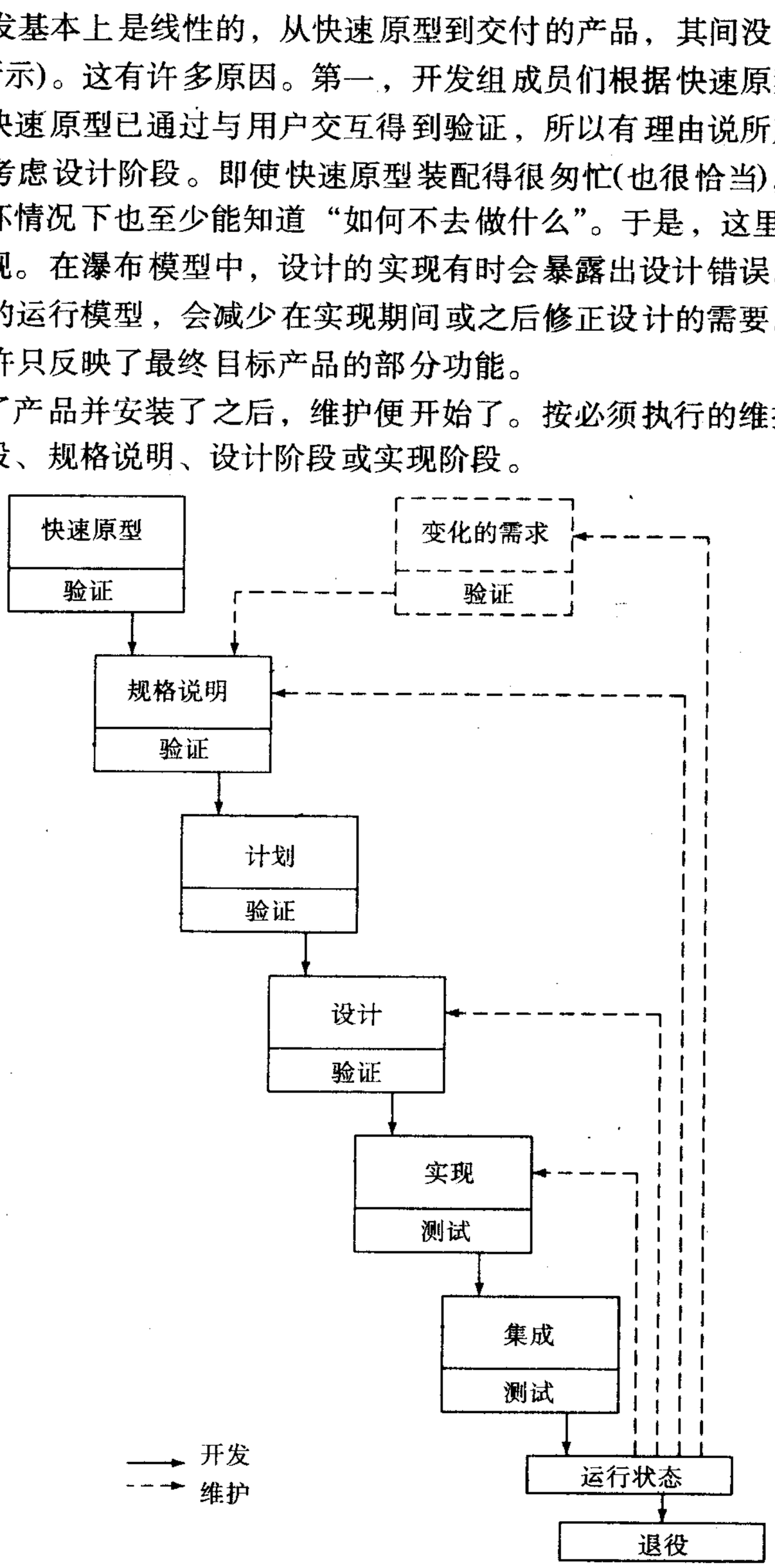

图3-3 快速原型模型

快速原型的本质体现在“快速”一词上。开发人员应尽可能快速地建造出快速原型，以加速软件的开发过程。毕竟，快速原型的唯一用途是判断客户的真正需求是什么，一旦需求确定了，快速原型将被抛弃。因此，快速原型的内部结构并不重要。重要的是，原型必须迅速地构建，然后迅速地对之进行修改，以反映客户需要，因此，速度是快速的原型的本质。

瀑布模型和快速原型模型的结合

尽管瀑布模型有许多成功之处，但也有重大的缺点：交付给客户的产品也许不是客户所真正需要的。快速原型也有许多成功之处。不过，如第7章所述，这种模型迄今还未完全经受考验，它可能也有自身的缺陷。

一种解决方案是将这两种方法结合起来，这种方法可通过对照图3-2瀑布模型的各阶段与图3-3快速原型模型的各个阶段来理解。快速原型可用作需求分析技术，换句话说，第一步建立一个快速原型，以便确定客户的真正需求，然后把该快速原型作为瀑布模型的入口。

这种方法具有有用的副作用。有些组织由于担心使用新技术而带来的风险，于是勉强地使用快速原型方法。将快速原型作为瀑布模型的先导，将使公司的管理部门有机会在最小的风险下评估一种新技术。

快速原型模型将在第7章讨论需求的过程中更详细地进行分析。现在考察另一类生命周期模型。

3.4　增量模型

软件是制造出来的，不是写出来的。这就是说，软件和一幢楼房一样，也是一步一步建造出来的。在软件产品开发过程中，每一步都是在前一步基础上又加了一步。某一天，设计做了扩充，而第二天，另一个模块编好了代码。整个产品的构建按这种递增方式不断向前，直到完成。

当然，并不是每天都在前进。正如一位建设者偶而必须拆散一面位置不正确的墙，或换一块被一个粗心的油漆工打破了的玻璃。软件开发人员有时必须对产品重新进行规格说明、重新设计、重新编码，或在最坏情况下，扔掉业已完成的产品而从头开始。但产品有时以间断式前进这一事实，并没有否定软件产品是一块一块制作的这一基本的事实。

软件开发的这种递增方式，导致开发以所谓的增量模型(incremental model)进行，如图3-4所示。产品被作为一系列的增量构件来设计、实现、集成和测试，每个构件是由多种相互作用的模块所形成的提供特定功能的代码片段构成。例如，如果产品要控制一般的核潜艇系统，则可能形成一个构件，武器控制系统可能也是一个构件。在一个操作系统中，调度程序可能是一个构件，文件管理系统可能也是一个构件。在增量模型的各个阶段，一个新的构件将被编码，然后并入一个软件体系结构中，并把该结构作为一个整体进行测试。这个过程直到产品达到目标功能，也就是产品满足了它的规格说明时，才会终止。开发人员可自由地按自己的方式把目标产品分解成构件，唯一的约束条件是当构件并入现有软件时，所形成的产品必须是可测试的。如果把产品分解为很少几个构件，则增量模型将退化成边做边改方法(如图3-1所示)，反之，如果产品由太多的构件组成，则在每个阶段，将有大量时间浪费在只是少量附加功能的测试上。最佳分解因产品而异、因人而异。

增量模型已取得了一些成功。例如，Wong在一份报告中说，她使用增量模型以及其他的现代软件开发技术，在25个月里，在预算之内，为空中防御系统开发了一套软件 [Wong,1984]。

下面分析增量模型的优点。

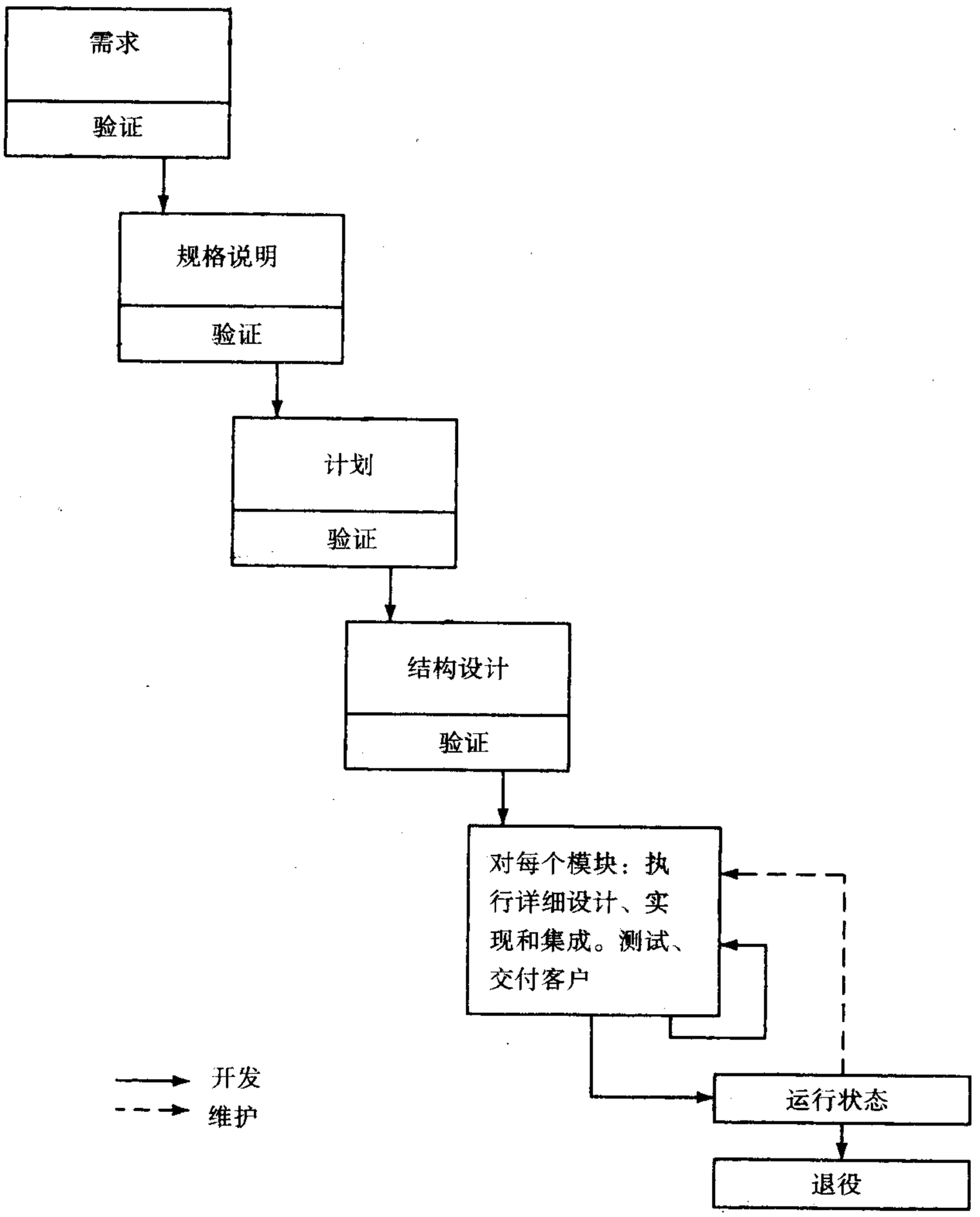

图3-4 增量模型

增量模型分析

瀑布模型和快速原型模型的目标都是开发一个完整的、可运行的产品，也就是要交付给客户一个满足所有需求且可投入运行的产品。正确地运用瀑布模型或快速原型模型，将使产品能得到完整的测试，使客户获得所需的功能。此外，这样的产品将附有充分的文档，这些文档不仅可在运行状态下使用，而且可以使按需求执行所有的三种维护(适应性、完善性和改正性维护)变得容易。这两种模型都有一个预定的交付日期，其意图是按预定的计划或提前交付整个产品。

相反，增量模型在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品。整个产品被分解成构件，开发人员一个构件接一个构件地交付产品。典型的产品通常由10到50个构件组成。每个阶段客户都得到一个完成部分需求的可运行产品。从第一个构件交付后开始，客户就能做有用的工作。使用增量模型，整个产品的各个部分可能在几周内就可使用了，而当使用瀑布模块或快速原型模型时，客户可能要等几个月甚至几年才会收到一个产品。

增量模型的另一个优点是，它减弱了给客户组织强加一个全新产品所带来的冲击。通过增量模型逐步地引入产品，使客户有时间适应新产品。变化是每个正在成长的组织的特性，而由

于软件产品是现实世界的模型，所以，需要变化也是一个软件的特性。变化和调整在增量模型中是自然的事，但当产品是以一步到位的方式开发和引入时，变化将是一种威胁。

从客户的资金流动看，分阶段交付不需要大量的资金输出。反而现金的流动将很合理，尤其是当前交付的构件能带来高额的投资回报时。与增量模型相关的优点在于，不一定要完成整个产品才能收回投资。相反，客户可在任何时候中止产品的开发。

使用增量模型的困难是，每个附加的构件在并入现有的软件体系结构时，必须不破坏原来已构造好的东西。此外，现有的结构必须便于按这种方式进行扩展，加入构件的过程必须简单、方便。尽管这种对开放结构的需要是一个短期的义务，但从长远看，开放结构是真正的优势。每个产品都首先经过开发，然后进入维护。在开发阶段，有一个清晰的规格说明和内聚性的设计的确很重要。但是，一旦产品进入维护阶段，对该产品的修改和大幅度增进的需求将很容易给内聚性的设计带来极大的破坏，使进一步的维护无法进行。在这种情况下，该产品要从头开发重建。设计必须是开放的，这一点不仅仅是增量模型的开发阶段所必不可缺少的，更是维护所必需的，不管开发阶段选择哪一种模型都是如此。因此，尽管增量模型比瀑布和快速原型模型需要更细心的设计，但将在维护阶段获得回报。如果一个设计非常灵活，足以支持增量模型，则这样的设计实际上允许在不破坏产品的情况下进行任何类型的维护。实际上，在使用增量模型时，开发一件产品和增进一件功能(维护)并没有区别，每次增进只是增加一个构件。

需求在开发过程中发生变化，这是太平常的事了，这一问题将在14.4.4节详细讨论。增量模型的灵活性使它应付这一问题的能力大大优于瀑布模型和快速原型模型。但另一方面，增量模型很容易退化为边做边改方法。这会使软件过程的控制丧失了整体性，最终的产品也不再是开放的，而是成为维护人员的恶梦。从某种意义上说，增量模型本身是自相矛盾的，它要求开发人员将产品视为一个整体，使设计从一开始就支持整个产品，包括未来的增进。但同时，它又要求开发人员将产品视为一个构件序列，每个构件本质上都独立于下一个构件。除非开发人员有足够的技巧来控制这一明显的矛盾，否则，按增量模型生产出来的产品可能不令人满意。

在图3-4所示的增量模型中，需求、规格说明、计划和结构化设计必须在开始实现各个构件前全部完成。增量模型的一种冒险的形式如图3-5所示。一旦客户的需求确定后，就开始草拟第一个构件的规格说明文档。完成后，规格说明组将转向第二个构件的规格说明，同时设计组开始设计第一个构件。这样，不同的构件将并行构建，而每个组都可利用从所有前面的构件中获取的信息。

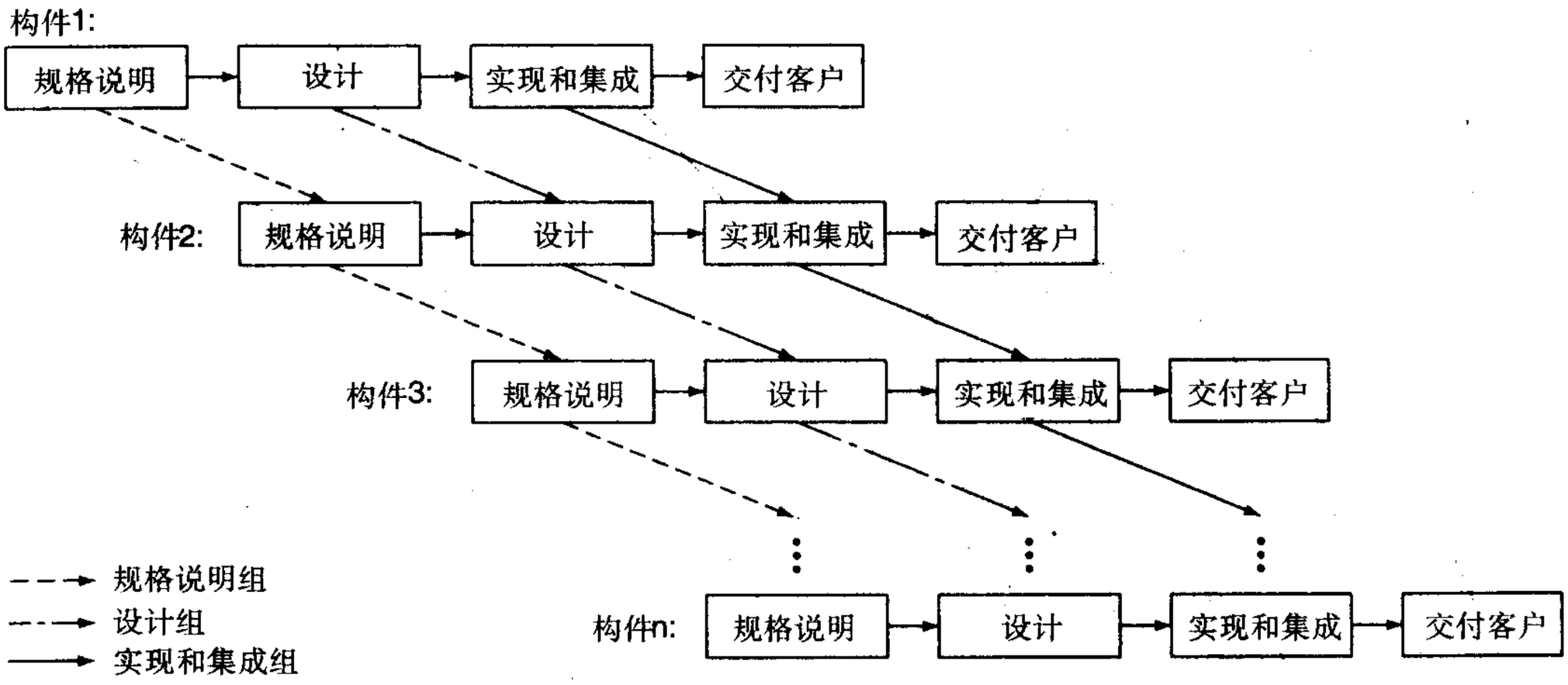

图3-5　更具风险的增量模型

这种方法会招致所做的构件将无法安装到一起的危险。在图3-4所示的增量模型中，由于开始做第一个构件之前，规格说明和结构设计必须已经完成，这意味着从一开始就已经有了总体设计。而在图3-5所示的模型中，除非对整个过程密切地监控，否则整个方案将有毁于一旦的危险。

因为螺旋模型结合了所有其他模型的特点，所以我们在最后介绍。

3.5 螺旋模型

软件开发几乎总是有一定的风险因素。例如，在产品还没有充分的文档之前，一些关键人员可能已经辞职。产品运行的关键硬件的生产厂家可能倒闭了。在测试和质量保证方面投资太多或投资太少。在花费了几十万美元开发一个大型产品后，新技术的突破使整个产品变得毫无价值。一家公司可能研制开发了一个数据库管理系统，但在产品投入市场之前，一个价格更低的、功能相近的软件包出台了。使用图3-5中的增量模型开发的产品各构件可能无法安装到一起。出于众所周知的原因，软件开发人员们只能努力尽可能地使这些风险降低到最低的限度。

一种可使某些风险降至最低的方法是构造一个原型。如3.3节所述，要降低交付的产品不满足客户的真正需要这一类风险，一种优秀的方法是在需求阶段构造一个快速原型。在其后的各阶段里，可以构造其他类型适合的原型。例如，一家电话公司可能要设计一个高效的算法来通过远程网络连接呼叫。如果为此实现一个产品，但产品并不像期望的那样好，则这家电话公司将浪费了开发产品的资金。此外，不满的顾客可能会将他们的生意转到其他公司。通过建立一个只控制呼叫接线的原型，并在模拟机器上测试它，就可避免这类情况的出现。使用这种方法，实际的系统将不被干扰，而且根据实现接线算法的费用，电话公司就能判断开发一个融入这个新算法的整个网络的控制程序是否值得。

通过使用原型和其他方法来使风险降至最小的思想，正是螺旋模型(spiral model)[Boehm,1988]的基础。看待这种生命周期模型的一种简便方法是把它看作一个瀑布模型，且每个阶段之前都有一个风险分析过程，如图3-6所示(该图的一部分在图3-7中重新画了，以反映术语螺旋模型的含义)。在每个阶段开始之前，都有一个控制(或排除)风险的尝试。如果在某个阶段不能排除所有重大的风险，方案将立即终止。

可有效地利用原型来搜集关于某些类风险的信息。例如，定时约束条件一般可通过构造一个原型并测量该原型是否能达到必需的性能来测试。如果原型的功能准确地代表了产品相关的特性，则对原型做的测量应该可以使开发人员对是否达到了定时要求有了很好的理解。

其他方面的风险是原型方法无能为力的。例如， 制作一个产品所必需的软件专业人员雇不到，或关键人员在方案完成前辞职了，这些风险很常见。另一个潜在的风险是，一个开发组可能没有足够的能力去开发某一个大规模产品。这类风险无法通过在一个更小的原型上测试他们的能力来解决。大规模软件和小规模软件之间有许多本质的区别，因此原型方法几乎没有用。另一类不能利用原型方法的风险是评估一家硬件销售商的交付承诺。开发人员可采用的策略之一是了解该销售商以前客户所受到的待遇如何，但过去的表现决不能一定代表未来的表现。在交付合同中加入惩罚条款是尽力保证必要的硬件能及时收到的方法之一，但是，如果销售商拒绝签订带惩罚条款的协议该怎么办？即使有惩罚条款，交付期推迟也可能会发生，最终导致有时长达几年的官司。而同时，软件开发商可能因为没有及时交付的硬件使软件不能按期交付而导致破产。简而言之，尽管原型方法有助于降低某些领域内的风险，但在其他领域，它最多只能有部分的用途，而在有些领域，则毫无用途。

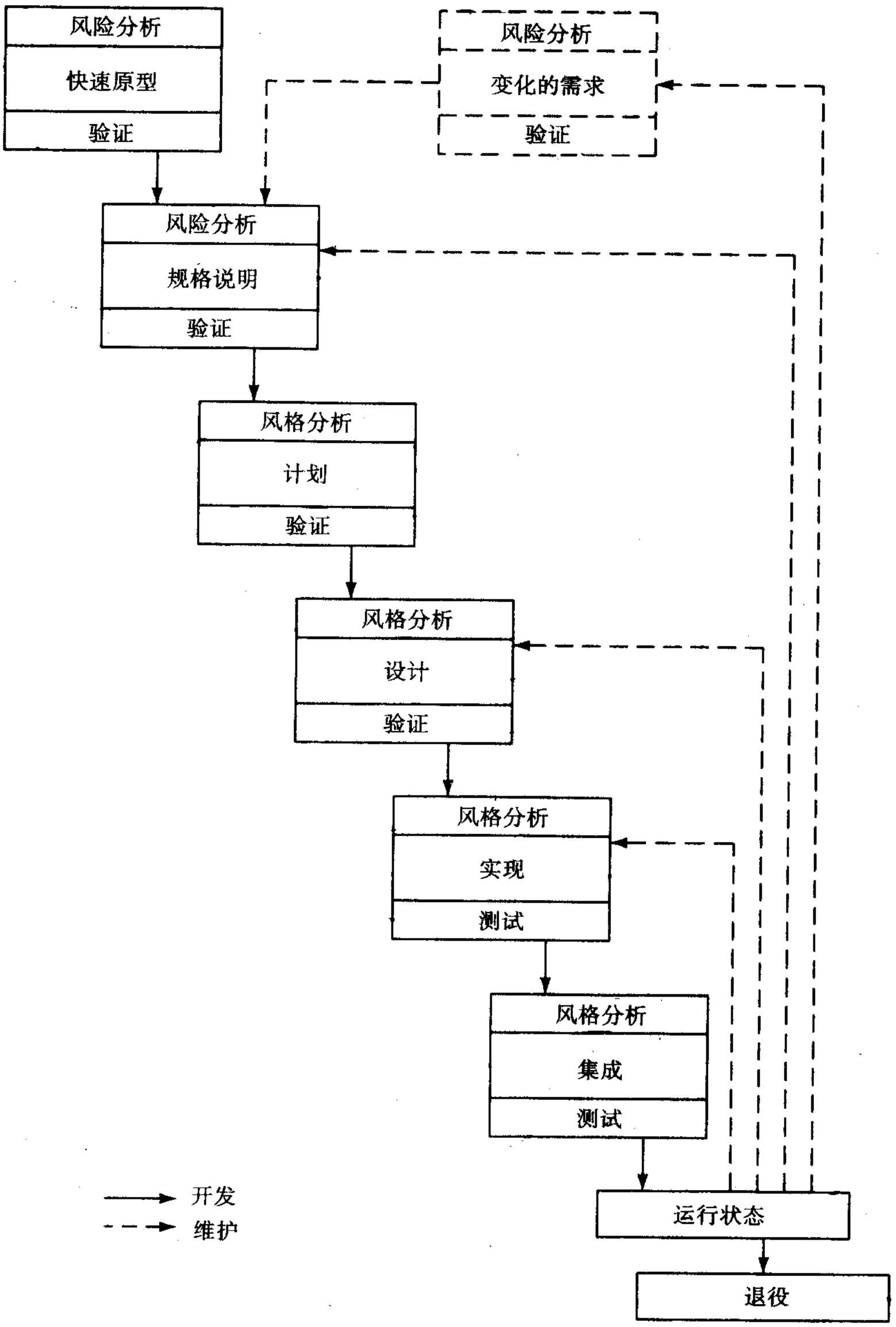

图3-6　螺旋模型的简化形式

完整的螺旋模型如图3-8所示。其中，带箭头的点划线长度代表当前的累计费用，角度值代表扫过螺旋的进度。螺旋的每个周期对应一个阶段。一个阶段开始(在上部1/4圆的部分)是确定该阶段的目标，完成这些目标的选择方案，以及施加于这些选择方案上的约束条件。这个过程以完成这些目标的策略为结果。再下一步，将从风险角度分析该策略，并努力排除各种潜在的风险，有些情况通过建造一个原型来完成。如果某些风险不能排除，方案立即终止，然而在某些环境下可能会决定继续实施此项目，但项目的规模将大大减少。如果所有的风险都成功地排除了，将启动下一个开发步骤(右下1/4圆部分)。螺旋模型的这个1/4圆部分相当于纯粹的瀑布模型。最后是评价该阶段的结果，并设计下一个阶段。

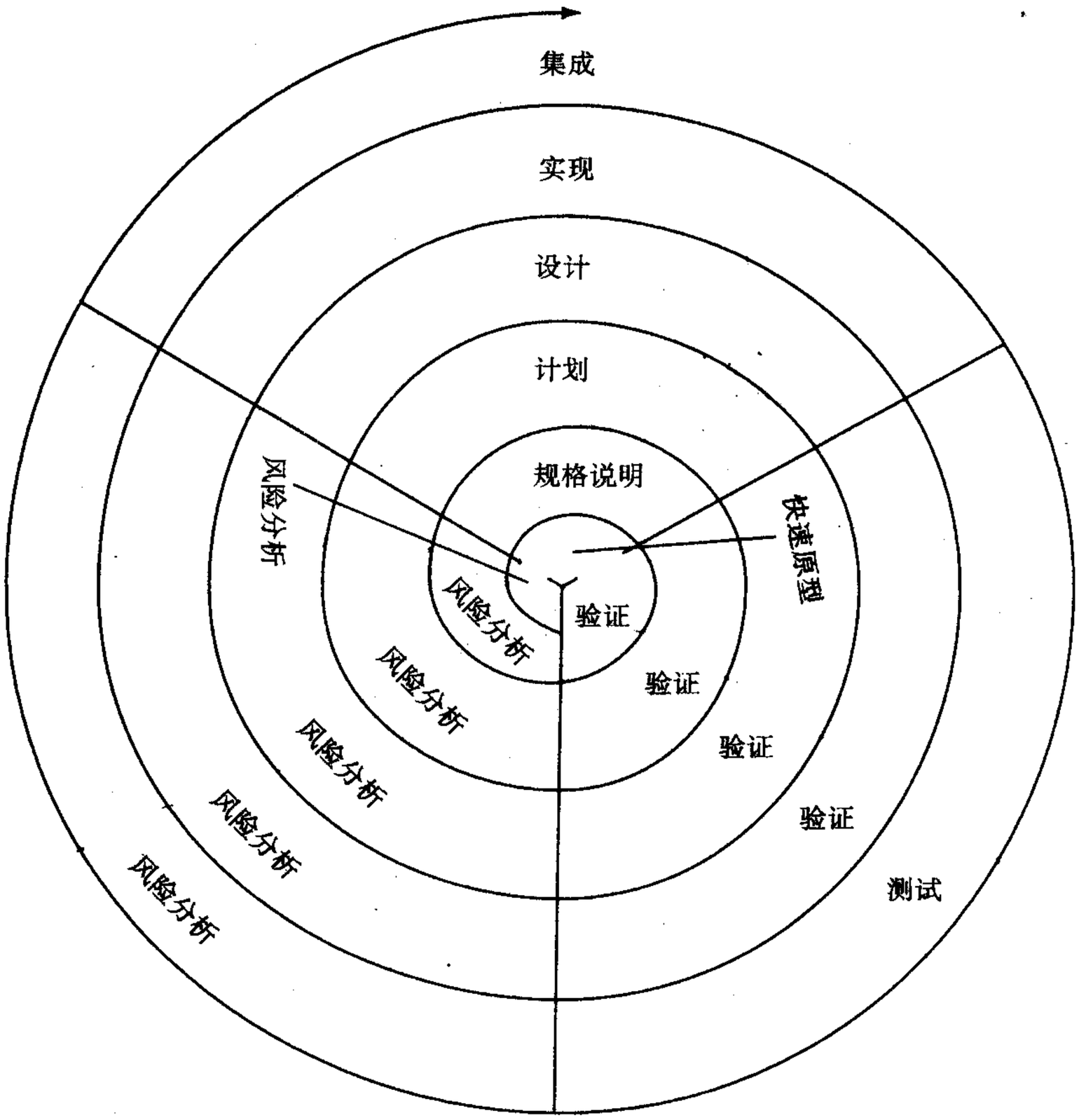

图3-7 按图3-6的一部分重画的螺旋模型

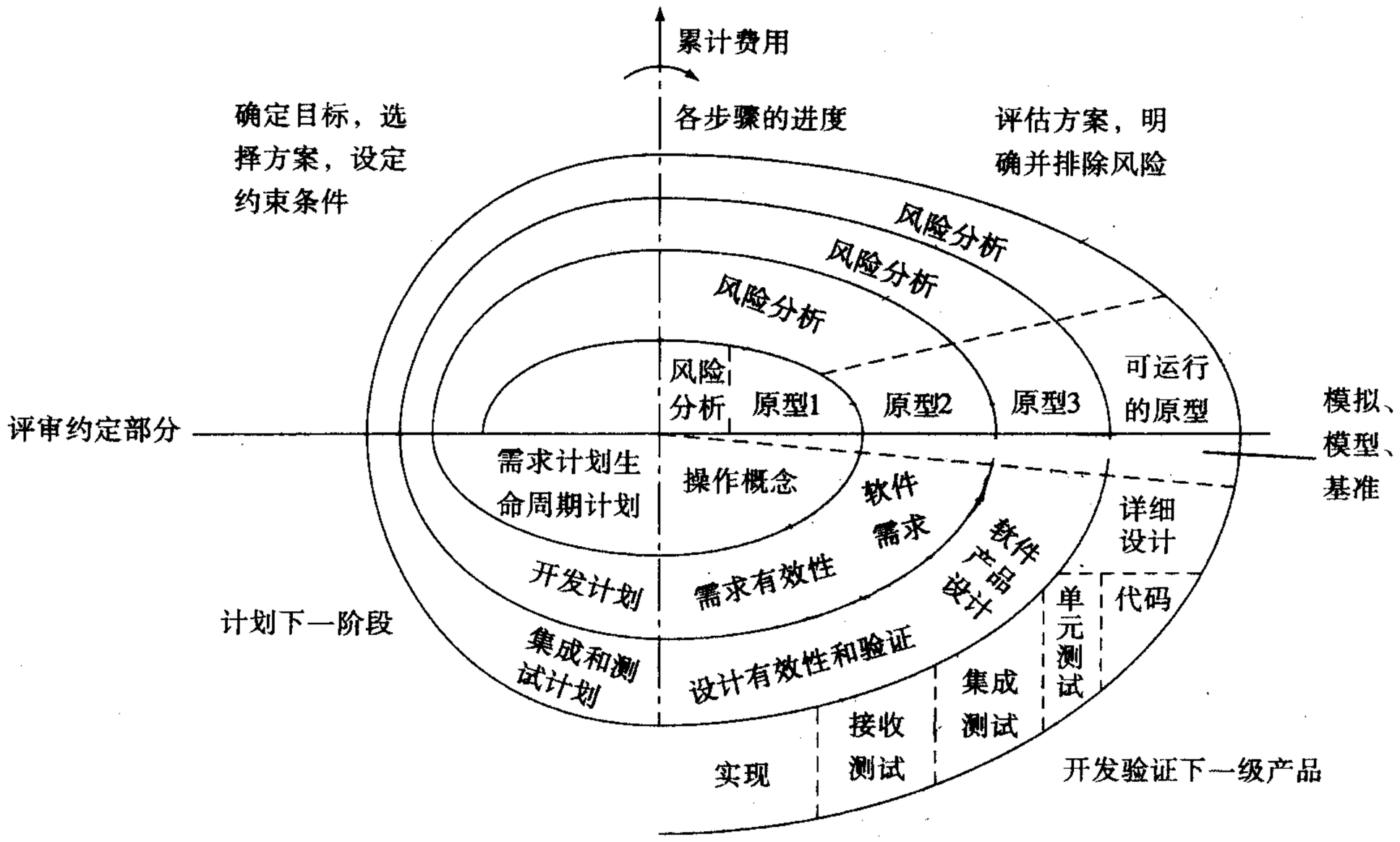

图3-8 完整的螺旋模型[Boehm,1988]。(@ 1988IEEE)

螺旋模型已在各种各样产品的开发中取得了成功。在25个用螺旋模型以及其他提高生产率的方法开发的项目中，每个项目的生产率比原来的生产水平提高了至少50%，其中大多数项目提高了100%[Boehm,1988]。为了判断一个给定的项目是否用螺旋模型，下面评价一下螺旋模型的优缺点。

螺旋模型分析

螺旋模型有许多优点。对可选方案和约束条件的强调支持了现有软件的重用(6.10节)，有助于将软件的质量作为一个特殊的目标融入产品开发中。此外，软件开发中的一个共同问题是判断一个特定的产品在何时已得到了充分的测试。花太多时间在测试上会浪费资金，导致产品交付期推迟。相反，如果执行的测试太少，则交付的产品可能包含许多残留故障，给开发人员带来不愉快的结果。螺旋模型解决了做太多测试或未作足够测试所带来的风险。也许最为重要的是，在螺旋模型结构中，维护只是螺旋模型的另一个周期；在维护和开发之间本质上并没有区别。因此，由于维护以一种和开发同等的方式被看待，使维护有时会遭到一些无知的软件专业人员的抵触的这一问题不复存在。

螺旋模型有一定的适用限制条件。特别是，当前形式的螺旋模型排它性地适用于内部的大规模软件开发[Boehm,1988]。考虑一个内部开发项目，即开发人员和客户是同一组织内成员的项目。如果风险分析的结果认为该项目应被终止，则只需要给这些软件人员重新分配一个项目。然而，一旦开发组织和一个外面的客户签定了合同，则哪一方面要中止这个合同都要承担违反合同的法律责任。因此，在开发合同软件时，所有的风险分析都必须在合同签署前，由客户和开发商一起进行，而不是像螺旋模型中那样。

第二个限制条件与项目的规模有关。特别是，螺旋模型只适用于大规模软件项目。如果执行风险分析的费用与整个项目的费用接近，或者如果执行风险分析会极大地影响利润潜力，则进行风险分析将毫无意义的。因此，开发人员应首先判断冒风险的代价有多少，然后判断执行风险分析的代价要多大。

螺旋模型的主要优势在于它是风险驱动的(risk-driven)，但这也可能是个弱点。除非软件开发人员擅长寻找可能的风险，准确地分析风险，否则将会有真正的危险：当项目实际上正在走向灾难时，开发组可能还认为一切良好。只有在开发组成员是合格的风险分析员的情况下，管理部门才能决定使用螺旋模型。

3.6　各种生命周期模型的比较

到目前为止已经描述了五类软件生命周期模型，并着重分析了它们的优缺点。瀑布模型广为人知，它的优缺点人们都知道。快速原型模型是为了解决瀑布模型中所发现的特定缺陷(即交付的产品可能不会是客户真正需要的)而开发的。人们对新的快速原型模型的了解要比对熟悉的瀑布模型的理解少得多。但快速原型模型可能有其自身的问题。一种选择是将这两种模型的优点结合起来，如3.3.1节所叙述。另一种选择是使用一种不同的模型，即增量模型。这种模型尽管有成功之处，但也有许多缺陷。还有一种选择是使用螺旋模型，但这种模型只有在开发人员在风险分析和风险排除方面受过充分培训的情况下才能使用。另一个必须考虑的因素是，当使用面向对象的范型时，软件生命周期模型必须是循环的，即它必须支持反馈。此外，增量方法似乎是更合适的。面向对象的生命周期模型将在第9章介绍。

每个软件开发组织都应该选择适合于该组织及其管理、职员、软件过程的软件生命周期模型，而且随着当前正在开发的特定产品的特性而变化。这样的模型将融入各种生命周期模型的合适的特性，以最小化它们的缺点，充分利用他们的优点。

3.7 能力成熟度模型

能力成熟度模型CMM(Capability Maturity Model)不是一个软件生命周期模型，而是一种改进软件过程的策略，它与实际使用的模型无关。CMM在1986年由美国卡内基-梅隆(Carnegie-Mellon)大学软件工程研究所SEI(Software Engineering Institute)的Watts Humpherey提出[Humpherey,1989] (要知道CMM的以前名称，请参见本章的“3-I 你想知道的进一步的信息”)。CMM的基本思想是，因为问题是由我们管理软件过程的方法引起的，所以新软件技术的运用不会自动提高生产率和利润率。CMM有助于组织建立一个有规律的、成熟的软件过程。改进过的过程将会生产出质量更好的软件，使更多的软件项目免受时间和费用的超支之苦。

软件过程包括各种活动、技术和用来生产软件的工具。因此，它实际上包括了软件生产的技术方面和管理方面。CMM策略力图改进软件过程的管理，而在技术上的改进是其必然的结果。

必须牢记，软件过程的改善不可能在一夜之间完成，CMM是以增量方式逐步引入变化的。CMM明确地定义了5个不同的“成熟度”等级，一个组织可按一系列小的改良性步骤向更高的成熟度等级前进。为理解这种方法，现在描述5种成熟度等级[Paulk,Curtis, Chrissis, and Weber,1993]。

3-I 你想知道的进一步的信息

当1986年将能力成熟度模型CMM初次引入时，它被称为过程成熟度模型。目前仍有一些组织仍使用这个老名称。不过，随着这些组织开始使它们的技术现代化，它们也开始使其软件术语现代化。

软件工程研究所有一个新的发展是个人软件过程PSP (Personal Software Praess)。PSP帮助软件工程师们改进他们的个人工作风格，而CMM的目标是组织范围内的改进初步成果的显示。使用PSP能提高质量和生产率[Humpherey, 1995]。

成熟度等级1：初始级(Initial)。处于这个最低级的组织，基本上没有健全的软件工程管理制度。每件事情都以特殊的方法来做。如果一个特定的工程碰巧由一个有能力的管理员和一个优秀的软件开发组来做，则这个工程可能是成功的。然而通常的情况是，由于缺乏健全的总体管理和详细计划，时间和费用经常超支。结果，大多数的行动只是应付危机，而非事先计划好的任务。处于成熟度等级1的组织，由于软件过程完全取决于当前的人员配备，所以具有不可预测性，人员变化了，过程也跟着变化。结果，要精确地预测产品的开发时间和费用之类重要的项目，是不可能的。

很不幸，世界上绝大多数的软件公司都处于成熟度等级1。

成熟度等级2：可重复级(Repeatable)。在这一级，有些基本的软件项目的管理行为、设计和管理技术是基于相似产品中的经验，故称为“可重复”。在这一级采取了一定措施，这些措施是实现一个完备过程所必不可缺少的第一步。典型的措施包括仔细地跟踪费用和进度。不像在第一级那样，在危机状态下方行动，管理人员在问题出现时便可发现，并立即采取修正行动，以防它们变成危机。关键的一点是，如没有这些措施，要在问题变得无法收拾前发现它们是不可能的。在一个项目中采取的措施也可用来为未来的项目拟定实现的期限和费用计划。

成熟度级3：已定义级(Defined)。在第3级，已为软件生产的过程编制了完整的文档。软件过程的管理方面和技术方面都明确地做了定义，并按需要不断地改进过程，而且采用评审的办法来保证软件的质量。在这一级，可引用例如CASE环境(4.3.1节)来进一步提高质量和产生率。

而在第一级过程中，“高技术”只会使这一危机驱动的过程更混乱。

尽管有些组织已达到了成熟级2和3，但迄今还没有一个组织达到了第4级和第5级，尽管有某个独立项目或某个小组达到了这两级。因此，这两个最高级是未来的目标。

成熟度级4：已管理级(Managed)。一个处于第4级的公司对每个项目都设定质量和生产目标。这两个量将被不断地测量，当偏离目标太多时，就采取行动来修正。利用统计质量控制[Deming.1986,Juran,1988]，管理部门能区分出随机偏离和有深刻含义的质量或生产目标的偏离(统计质量控制措施的一个简单例子是每千行代码的错误率。相应的目标就是随时间推移减少这个量)。

成熟度级5：优化级(Optimizing)。一个第5级组织的目标是连续地改进软件过程。这样的组织使用统计质量和过程控制技术作为指导。从各个方面中获得的知识将被运用在以后的项目中，从而使软件过程融入了正反馈循环，使生产率和质量得到稳步的改进。

这5个成熟度等级在表3-1中作了总结。要改进其软件过程，一个组织首先应尽量对它当前的过程有一定的了解，然后明确它期望的过程。第2步是确定完成这个过程改进所要采取的行动，并按优先顺序排列。最后，拟定一个实现这一改进的计划并付诸实施。这一系列步骤在下一次改进中将再重复进行。

表3-1　能力成熟度模型的5等级

成熟度等级	特　点
1. 初始级	特殊的过程
2. 可重复级	基本的方案管理
3. 已定义级	过程定义
4. 已管理级	过程措施
5. 优化级	过程控制

关于能力成熟度模型方面的经验表明，提高一个完整的成熟度等级通常需要花18个月到3年的时间，但从第1级上升到第2级有时要花3年甚至5年时间。这反映了要向一个迄今仍处于特殊的和被动的行动方式的公司慢慢地灌输系统的方式，将是多么的困难。

对每个成熟度等级，SEI着重给出了一系列关键的过程方面，以使一个组织在向下一个成熟度等级提升时有了目标。每个过程方面是一组相关的目标，如果完成这些目标，就能达到下一个成熟级。例如对于成熟度等级2，关键的过程方面包括软件质量保证(5.1.1节)、配置管理(4.6节)和项目计划(第10章)。在最后的一个方面里，目标之一是制定一个适当的、覆盖了所有软件生产行动的计划。在最高的第5级，关键的行为包括故障预防、技术革新和过程变化管理。比较这两级的目标就可清晰地看出，第5级组织比第2级组织要领先许多。例如，第2级组织关心软件质量保证，即关心故障的识别和修正(软件质量将在第5章详细讨论)。相反，第5级组织的软件过程中融入了故障预防，即首先努力保证软件中没有故障。

为帮助各组织达到更高成熟度等级，SEI已开发了一系列征求意见的表(成熟度提问单)，作为SEI成熟试等级评估的基础。评估的目标是突出一个组织软件过程中的目前缺点，并指出该组织改进其软件过程的方法。

SEI主要由美国国防部(DoD)投资。CMM计划的最初目标之一是评估为DoD生产软件的承包商的软件过程，并将合同给那些过程较成熟的承包商，以此来提高国防软件的质量。然而，CMM计划已远远超过了改进DoD软件过程这一目标，正在被众多希望提高软件质量和生产率的软件组织贯彻执行。

实现CMM能提高利润率吗？初步的成果表明的确如此。例如，加利福尼亚州Fullerton的Hughes Aircraft的软件工程部在1987年～1990年间，在评估和改进计划上花了大约500 000美元

[Humphrey,Snider,and Willis,1991]。在这3年期间，Hughes Aircraft从成熟度等级2上升到成熟度等级3，并致力于向第4级甚至第5级迈进。作为软件过程改进的结果，Hughes Aircraft估计每年节约了大约2百万美元。这些节约是通过许多途径产生的，包括加班小时的减少、更少的危机、职员技术的提高和更少的软件人员调整。

而其他组织也报告了类似的成果。例如，位于Raytheon的设备部从1988 年的成熟度的第1级上升到1993年的第3级。其结果是生产率提高两倍，且投资在过程改进工作方面的资金每1美元收益了7.7美元 [Dion,1993]。Shlumberger取得了类似的成功[Wohlwend and Rosenbaum,1993]。由于这些成就，CMM已开始在软件工业界得到广泛的运用。

Rifkin运用来自于1300个项目中的数据，制作了一个模型，得出了当同一个200 000行的数据处理产品分别由处于第1级到第5级的公司开发时期望的故障数和总费用[Rifkin,1993]。模型的预测如表3-2所示。表中显示了一个组织从第1级上升到第5 级开发该项目所需要的费用，可通过软件质量的提高和开发费用的大幅度降低而得到补偿，且绰绰有余。

表3-2 当一个200 000行的数据处理产品由处于CMM第1级到第5级的组织开发时的模型预测(版权1993, Master Systems有限公司)

CMM级	持续时间 日历月数	工作人月数	交付给客户和 安装发现的故障	开发中发现 的故障	开发的 总费用
第1级	29.8	593.5	61	1 348	$5 440 000
第2级	18.5	143.0	12	328	$1 311 000
第3级	15.2	79.5	7	182	$728 000
第4级	12.5	42.8	5	97	$392 000
第5级	9.0	16.0	1	37	$146 000

CMM越来越重要还因为另一个原因。美国空军规定，任何希望成为空军承包商的软件开发组织必须到1998年达到CMM第3级。也许整个美国国防部不久也将发布类似指示。因此，软件组织有改进其软件过程成熟度等级的压力。

3.8 ISO 9000

如前所述，SEI的能力成熟度模型(CMM)试图通过改进软件的开发过程来改进软件质量。术语“成熟度”本质上是过程本身优良程度的一种衡量。改进软件质量的另一种尝试是基于国际标准化组织(ISO)9000系列标准。

ISO 9000是由5个相关的标准系列组成，它适用于各种范围广泛的工业活动，包括设计、开发、生产、安装和服务；ISO 9000不仅仅是软件标准。在ISO 9000系列中，为质量系统而制定的ISO 9001标准[ISO 9001,1987]是最适用于软件开发的标准。鉴于ISO 9001的广泛性，ISO公布了特殊的纲领，以有助于ISO 9001运用于软件，此项纲领就是ISO 9000-3 [ISO 9000-3,1991]。

ISO 9000有许多不同于CMM[Dawood,1994]的特征。ISO 9000 强调要用文字和图形对过程进行文档的编写，以保证一致性和可理解性。ISO 9000的基本原理是，坚持标准不能100%地保证产品的质量，但能降低产品品质较差的危险。ISO 9000只是质量系统的一部分。同样需要的是管理部分的质量承诺，密切关注工作人员的培训，以及为持续的质量改进确定目标并实现。

和CMM一样，ISO 9000也强调度量。两种模型都强调确保过程改进所必须采取的修正行动。最后，两者都着重强调必须不断地培训软件专业人员，而不仅仅在引入质量改进措施时才进行培训。

ISO 9000系列标准已得到了60多个国家的采用，包括美国、日本、加拿大和欧共体各国。这意味着，如果一家美国的软件组织同欧洲的客户做生意，那么这家美国的组织首先必须被确认符合ISO 9000标准。验证登记员(审查员)必须考查该公司的软件过程，证明该过程与ISO标准一致。

虽然CMM和ISO 9000有相同的目标，即改进软件质量，但当判断成熟度等级或是否与ISO 9000一致时，两者将分别考虑不同的属性。因此，即使两者在主要方面重叠，但据报导，至少有两个第1级组织也被确认与ISO 9000一致[Bishop,1994]。相反，在理论上，也许一家显示出高级成熟度的组织也可能不符合ISO 9000[Bamford and Deibler,1993a]。然而，一般来说，一个成熟的组织的软件过程符合ISO 9000的可能性非常大。

跟欧洲一样，越来越多的美国公司正在申请ISO 9000认证。例如，General ElectricPlastic Division要求340家厂商在1993年7月之前达到这一标准[Dawood,1994]。美国政府不可能跟着欧洲的领导而要求想和美国做生意的外国公司必须获得ISO 9000认证。不过，对美国及其贸易伙伴也具有一定的压力，这些压力可能最终导致ISO 9000成为世界范围内的标准。

本章回顾

本章描述了许多不同的过程模型。边做边改模型(3.1节)应避免使用。瀑布模型(3.2节)的优势在于它是规则的、文档驱动的方法。这种模型的问题是，交付的产品可能不是客户真正需要的。快速原型模型(3.3节)正是为解决这一问题而开发的。增量模型(3.4节)具有能在早期使投资获得极大的回报和易于维护的优点，但也有其本身困难，尤其是对开放结构的需要。风险驱动的螺旋模型适用于大规模的内部开发的项目(3.5节)。3.6节中建议，不同软件生命周期模型的特征应该结合起来，以提出一个适当的生命周期模型。该模型的形式将取决于本组织，它的管理、职员及要构造的产品。最后，在3.7节介绍了能力成熟度模型，3.8节介绍了ISO 9000。

进一步阅读

[Boehm,1981]的第4章中描述了瀑布模型，这一章还包括一个与本章3.4节不同的增量模型实例。

要了解快速原型，建议阅读这三本书：[Lantz, 1985]、[Connell and Shafer, 1989] 和[Gane, 1989]。计算机辅助的原型方法的作用在[Luqi and Royce,1992]中做了评价。

[Gilb, 1988]中介绍了另一种形式的增量模型，即演进式交付模型。另一种增量模型在[Currit, Dyer and Mills, 1986]和[Selby,Basili,and Baker, 1987] 中做了描述。[Boehm, 1988] 解释了螺旋模型， 而该模型在TRW 软件生产系统中的运用参见[Boehm, Penedo, Stuckle, Williams,and Pyster, 1984]。风险分析在[Charette, 1989]和[Boehm, 1991]中介绍。

许多其他的生命周期模型也被提出。一种强调人的因素的软件生命周期模型在[Mantei and Teorey,1988] 中给出。NASA软件工程实验室推荐的生命周期模型在[Landiset al.,1992]中描述。面向对象的生命周期模型在[Henderson-Sellers and EdWards,1990],[Booch,1994]和[Rajlich,1994]中介绍。一种比较可选模型的方法在[Davis Bersoff, and Comer,1988]中给出。国际软件过程工作站的学报是一个有用的生命周期模型信息源。

SEI能力成熟度模型(CMM)在 [Humphrey, 1989] 中有更详细的介绍。一个新的版本在[Paulk Curtis, Chrissis, and Weber, 1993] 和 [Paulk etal, 1993] 中描述。CMM的成功在[Humphery, Snider, and Willis, 1991]和[Hicks and Card, 1994]中描述，而关于评估过程的评论出现在[Bollinger and McGowan,1991]中。1993年7月版的IEEE Software中包含了许多关于CMM

论文。对在1987年和1991年间SEI软件过程评估的评价在[Kitson and Masters, 1993] 中给出，它对软件工业的影响在[Saiedien and Kuzara, 1995] 中讨论。[Bamford andDeibler, 1993b]中详细地比较了ISO 9000和CMM，而概述出现在[Bamford and Deibler,1993a]中，[Paulk,1995]中也做了这类比较。1993年12月版的IEEE Transaction onSoftware Engireering和1994年7月的IEEE Software中包含许多关于过程改进的论文。[Fuggetta and Picco,1994]是另一个关于过程的注释性文献。

问题

3.1 假设你要做一个给73 624.9385开平方，并精确到小数点后四位这样的产品。一旦实现并测试完成后，该产品将被扔掉，你将使用什么模型？说出你回答的理由。

3.2 你是一个软件工程顾问。一家生产和销售长统靴的公司负责财政的副总裁召见了你。她要你们的组织为她公司生产一个产品，该产品将监控该公司的存货。负责跟踪首先是购买橡胶，再进行靴子的生产、发送到各个商店，以及卖给顾客的整个过程。你在为该项目选择生命周期模型时将使用什么准则？

3.3 列出在开发问题3.2的产品中会遇到的风险。你怎样尝试排除这些风险？

3.4 你为靴类连锁店开发的存货控制产品是如此的成功，以至于你的组织决定必须将它重新写成软件包，以卖给各种生产并通过他们自己的销售商店销售产品的组织。这个新产品因此必须是可移植的，并能很容易适应新硬件或新的操作系统。你在选择模型时将使用哪些不同于你在回答问题3.2时的准则？

3.5 说明哪类产品最适合增量模型。

3.6 现在描述一下增量模型可能导致困难的情形。

3.7 说明哪一类产品最适合用螺旋模型？

3.8 现在描述一下螺旋模型不适用的情形。

3.9 你被Olde Fashioned Software Developers公司雇佣，帮助改进他们的生产率和利润。你很快发现，该公司毫无疑问是处于成熟级1。你采取的第一个步骤是什么？

3.10 (学期项目) 针对附录A中的Osbert Oglesby项目，你将采用哪种生命周期模型？说出你回答的理由？

3.11 (软件工程文献阅读) 你的导师将让你们阅读[Boehm,1988]。说明在螺旋模型中融入瀑布模型和快速原型模型的特征的情况。这两种模型中哪一种占优势？