第3章

软件开发过程管理

3.1 CMM 和 ISO 9000

软件过程是指人们用于开发和维护软件及其相关产品的一系列活动、方法、实践和革新。软件开发过程管理是指在软件开发过程中，除了先进技术和开发方法外，还有一整套的管理技术。这套管理技术研究如何有效地对软件开发项目进行管理，以便于按照进度和预算完成软件项目计划，实现预期的经济效益和社会效益。而软件过程改进则是针对软件生产过程中会对产品质量产生影响的问题而进行的，它的直接结果是软件过程能力的提高。现在常见的软件过程改进方法有：ISO 9000、SW-CMM 和由多种能力模型演变而来的 CMMI。本节将分别介绍 ISO 9000、SW-CMM 和 CMMI，并给出三者之间的异同之处。

3.1.1 SW-CMM 和 CMMI

为了保证软件产品的质量，1991 年美国卡内基-梅隆大学软件工程研究所（CMU/SEI）将软件过程成熟度框架进化为软件能力成熟度模型（Capability Maturity Model for Software，SW-CMM），并发布了最早的 SW-CMM 1.0 版。SW-CMM 为软件企业的过程能力提供了一个阶梯式的进化框架，阶梯共有 5 级，如图 3-1 所示。

1. 初始级

初始级的软件过程是未加定义的随意过程，项目的执行是随意甚至是混乱的。也许，有些企业制定了一些软件工程规范，但若这些规范未能覆盖基本的关键过程要求，且执行没有政策、资源等方面的保证时，那么它仍然被视为初始级。

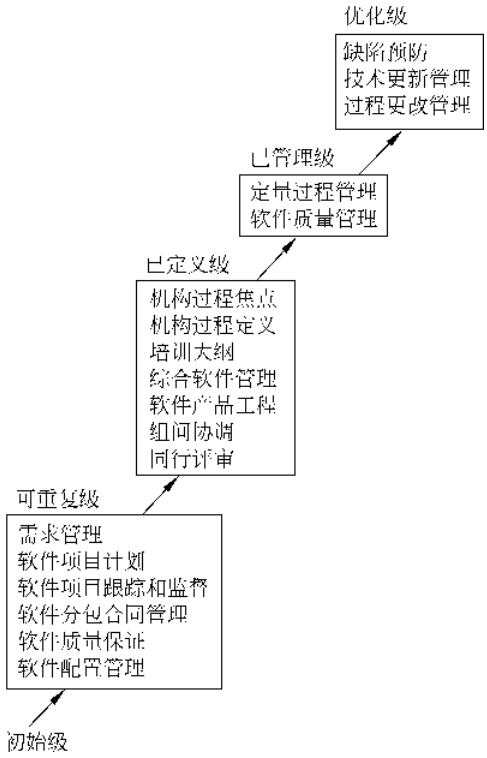

图 3-1　SW-CMM 示意图

2. 重复级

根据多年的经验和教训，人们总结出软件开发的首要问题不是技术问题而是管理问题。因此，第 2 级的焦点集中在软件管理过程上。一个可管理的过程是一个可重复级的过程，一个可重复级的过程则能逐渐进化和成熟。第 2 级的管理过程包括了需求管理、项目管理、质量管理、配置管理和子合同管理 5 个方面。其中项目管理分为计划过程和跟踪监控过程两个过程。通过实施这些过程，从管理角度可以看到一个按计划执行的且阶段可控的软件开发过程。

3. 定义级

在第 2 级仅定义了管理的基本过程，而没有定义执行的步骤标准。在第 3 级则要求制定企业范围的工程化标准，而且无论是管理还是工程开发都需要一套文档化的标准，并将这些标准集成到企业软件开发标准过程中去。所有开发的项目需根据这个标准过程，剪裁出该项目的过程，并执行这些过程。过程的剪裁不是随意的，在使用前需经过企业有关人员的

批准。

4. 管理级

第 4 级的管理是量化的管理。所有过程需建立相应的度量方式，所有产品的质量（包括工作产品和提交给用户的产品）需有明确的度量指标。这些度量应是详尽的，且可用于理解和控制软件过程和产品，量化控制将使软件开发真正变成为工业生产活动。

5. 优化级

第 5 级的目标是达到一个持续改善的境界。所谓持续改善是指可根据过程执行的反馈信息来改善下一步的执行过程，即优化执行步骤。如果一个企业达到了这一级，那么表明该企业能够根据实际的项目性质、技术等因素，不断调整软件生产过程以求达到最佳。

除第一级外，SW-CMM 的每一级都是按完全相同的结构组成的。每一级包含了实现这一级目标的若干关键过程域（KPA），每个 KPA 进一步包含若干关键实施活动（KP），无论哪个 KPA，它们的实施活动都统一按 6 个公共属性进行组织，即每一个 KPA 都包含 6 类 KP。

（1）目标。每一个 KPA 都确定了一组目标，若这组目标在每一个项目都能实现，则说明企业满足了该 KPA 的要求。若满足了一个级别的所有 KPA 要求，则表明达到了这个级别所要求的能力。

（2）实施保证。实施保证是企业为了建立和实施相应 KPA 所必须采取的活动，这些活动主要包括制定企业范围的政策和明确高层管理的责任。

（3）实施能力。实施能力是企业实施 KPA 的前提条件，实施能力一般包括资源保证、人员培训等内容。企业必须采取措施，在满足了这些条件后，才有可能执行 KPA 的执行活动。

（4）执行活动。执行活动描述了执行 KPA 所需要的必要角色和步骤。在 6 个公共属性中，执行活动是唯一与项目执行相关的属性，其余 5 个属性则涉及企业 CMM 能力基础设施的建立。执行活动一般包括计划、执行的任务、任务执行的跟踪等。

（5）度量分析。度量分析描述了过程的度量和度量分析要求。典型的度量和度量分析的要求是为了确定执行活动的状态和执行活动的有效性。

（6）实施验证。实施验证是验证执行活动是否与建立的过程一致。实施验证涉及管理的评审和审计以及质量保证活动。

在实施 CMM 时，可以根据企业软件过程存在问题的不同程度确定实现 KPA 的次序，然后按所确定次序逐步建立、实施相应过程。在执行某一个 KPA 时，对其目标组也可采用逐步满足的方式。过程进化和逐步走向成熟是 CMM 体系的宗旨。

由于不同领域能力成熟度模型存在不同的过程改进，重复的培训、评估和改进活动以及活动不协调等一些问题，于是由美国国防部出面，美国卡内基-梅隆大学软件工程研究所

(CMU/SEI)于 2001 年 12 月发布的 CMMI 1.1 版本包括 4 个领域：软件工程(SW)、系统工程(SE)、集成的产品和过程开发(IPPD)、采购(SS)。

CMMI 有两种不同的实施方法，不同的实施方法，其级别表示不同的内容。CMMI 的一种实施方法为连续式，主要是衡量一个企业的项目能力，企业在接受评估时可以选择自己希望评估的项目来进行评估。而另一种实施方法为阶段性，它主要是衡量一个企业的成熟度，亦即是企业在项目实施上的综合实力。在这里简要介绍一下 CMMI 的 5 个台阶：

(1) 完成级。在完成级水平上，企业对项目的目标与要做的努力很清晰，项目的目标得以实现。但是由于任务的完成带有很大的偶然性，企业无法保证在实施同类项目的时候仍然能够完成任务，项目实施对实施人员有很大的依赖性。

(2) 管理级。在管理级水平上，企业在项目实施上能够遵守既定的计划与流程，有资源准备，权责到人，对相关的项目实施人员有相应的培训，对整个流程有监测与控制，并与上级单位对项目与流程进行审查。企业在二级水平上体现了对项目的一系列的管理程序，这一系列的管理手段排除了企业在一级时完成任务的随机性，保证了企业的所有项目实施都会得到成功。

(3) 定义级。在定义级水平上，企业不仅能够对项目的实施有一整套的管理措施，保障项目的完成，而且，企业能够根据自身的特殊情况以及自己的标准流程，将这套管理体系与流程予以制度化。这样，企业不仅能够在同类的项目上得到成功的实施，在不同类的项目上一样能够得到成功的实施。科学的管理成为企业的一种文化，成为企业的组织财富。

(4) 量化管理级。在量化管理级水平上，企业的项目管理不仅形成了一种制度，而且要实现数字化的管理，对管理流程要做到量化与数字化。通过量化技术来实现流程的稳定性，实现管理的精度，降低项目实施在质量上的波动。

(5) 优化级。在优化级水平上，企业的项目管理达到了最高的境界。企业不仅能够通过信息手段与数字化手段来实现对项目的管理，而且能够充分利用信息资料，对企业在项目实施的过程中可能出现的次品予以预防，能够主动地改善流程，运用新技术，实现流程的优化。

由上述的 5 个台阶可以看出，每一个台阶都是上面一阶台阶的基石。要上高层台阶必须首先踏上较低一层台阶。企业在实施 CMMI 的时候，应该要遵循循序渐进的原则，一般先从 2 级入手，逐步改进软件开发过程，争取最终实现 CMMI 的第 5 级。

3.1.2　ISO 9000 质量标准

所谓“ISO 9000”不是指一般意义上的一个质量保证标准，而是一族系列标准的统称。ISO 制定出来的国际标准除了有规范的名称之外，还有编号，编号的格式是：ISO＋标准号＋[杠＋分标准号]＋冒号＋发布年号(方括号中的内容可有可无)，例如 ISO 8402：1987、

ISO 9000-1：1994，分别是某一个标准的编号。

ISO 9000 的作用如下：①强化品质管理，增强客户信心，提高企业效益；②获得了国际贸易“通行证”，消除了国际贸易壁垒；③节省了第二方审核的精力和费用；④在产品品质竞争中处于有利的地位；⑤有效地避免产品质量问题；⑥有利于国际间的经济合作和技术交流。

中国软件企业之所以要进行 ISO 9001 质量体系认证，主要是出于两点考虑：一是参与国际竞争的需要，如果在管理上（首先是质量管理）不能和国际接轨，那么就会连参与竞争的资格都没有；二是为了引进先进的管理理论和方法，提高企业的管理水平，最终提高企业的综合竞争实力，使客户受益、股东受益、员工受益、社会受益。其中的第二点应当是进行认证的基本出发点，因为如果不着眼于从根本上提高企业的管理水平，就不可能真正和国际接轨，也就不可能真正参与国际竞争。

3.1.3 三者之间的比较

CMMI 被看作是把各种 CMM 集成为一个系列的模型。其与 SW-CMM 最大的不同点在于：

（1）CMMISM-SE/SW/IPPD/SS 1.1 版本有 4 个集成成分。

（2）SW-CMM 2 级共有 6 个关键过程区域（KPA），在 CMMI 中增加了一个：“度量和分析”。原来的 6 个关键过程区域的名称和内容在 CMMI 中作了部分改进，但是主体内容没有大幅调整。SW-CMM 4 级共有两个关键过程区域，在 CMMI 中仍是两个，只是名称和内容有所改进。SW-CMM 5 级共有 3 个 KPA，在 CMMI 中进行了合并，改为两个，但主要内容未变。变化最显著的在 SW-CMM 3 级与 CMMI 3 级之间，原有的 7 个 KPA 变成了 14 个，CMMI 对工程活动进行要求的 KPA——原 SW-CMM 3 级中的“软件产品工程”进行了详细的拆分，并结合常见的软件生命周期模型进行了映射。CMMI 中新增的关键过程区域中还涉及过去未曾提到的内容，例如“决策分析和解决方案”、“集成化群组”等，如表 3-1 所示。

表 3-1 SW-CMM 和 CMMI 的 KPA 比较

级别	SW-CMM 过程域	CMMI 过程域	说明与比较
2	需求管理 软件项目规划 软件项目追踪与监控 软件子合同管理 软件质量保证 软件配置管理	需求管理 项目计划 项目监督和控制 供应商合同管理 过程和产品质量管理 配置管理 度量和分析	项目过程管理的基本内容，实际是组织中某个或某几个团队的过程能力，可以说是 TSP 中的内容。这一级不同的是，在 CMMI 中增加了一个过程域“度量和分析”

续表

级别	SW-CMM 过程域	CMMI 过程域	说明与比较
3	软件过程要点 软件过程定义 培训计划 软件集成管理 软件产品工程 组间协作 同级评审	组织级过程焦点 组织级过程定义 组织级培训 集成化群组 集成化项目管理 组织级集成环境 集成供应商管理 需求开发 技术解决方案 产品集成 验证 确认 风险管理 决策分析和解决方案	开始从团队过程能力提升为组织过程能力，有关组织的过程域比较多，如“组织级过程定义和焦点”、“组织级培训”和“集成环境”、“产品集成”、“集成化项目管理”等；同时，也包含一些深层次的项目管理能力，如“需求开发”、“风险管理”、“决策分析和解决方案”等。SW-CMM 中“软件集成管理”，在 CMMI 中被分解为 4 个过程域——“集成化群组”、“集成化项目管理”、“集成供应商管理”和“组织级集成环境”。SW-CMM 中“软件产品工程”，在 CMMI 中被分解为 5 个过程域——“需求开发”、“技术解决方案”、“产品集成”、“验证”和“确认”。SW-CMM 的“组间协作”，包含在 CMMI 的“集成化项目管理”中，而 SW-CMM 的“同级评审”，包含在 CMMI 的“验证”中
4	过程量化管理 质量管理	项目定量管理 组织级过程性能	“组织级过程性能”是建立在“项目定量管理”的基础之上的，两者构成了一个完整的量化管理。SW-CMM 中的“质量管理”，被移到 CMMI 的 2 级中，发生了较大变动
5	缺陷预防 技术更改管理 过程更改管理	因果分析和解决方案 组织级改革和实施	“因果分析和解决方案”是通过对过程中出现的方法技术问题等进行分析，找出根本原因以解决问题，是战术性的改进；而“组织级改革和实施”是战略性的改进，组织的变革首先是管理文化的变革、质量方针和培训体系的变革等。而在 SW-CMM 中，主要强调在技术和过程两个方面进行持续改进。“缺陷预防”属于质量保证或管理中的基本内容，不应该放在第 5 级，所以 CMMI 做了修正

到底是选择 SW-CMM 还是 CMMI 主要考虑以下几个方面的因素：

① 实施企业的业务特点。

② 实施企业对过程改进的熟悉程度。

③ 实施企业对过程改进项目的预算。

④ 实施企业是否可以使用阶段式的演进路线。

⑤ 实施 CMM 与 CMMI 可以平滑地转换。

ISO 9001 与 CMM 的关系如下：

① ISO 9001 和 CMM 既有区别又相互联系，两者不可简单地互相替代。它们的最大相似之处在于两者都强调了“该说的要说到，说到的要做到”。CMM 强调的是持续过程改进，而 ISO 9001 的 1994 版标准主要说明的是“合格质量体系的最低可接受水平”(ISO 9001 的 2000 版标准也增加了持续改进的内容)。取得 ISO 9001 认证对于取得 CMM 的等级证书是有益的；反之亦然。

② 取得 ISO 9001 认证并不意味着完全满足 CMM 某个等级的要求。取得 ISO 9001 认证所代表的质量管理和质量保证能力的高低与审核员对标准的理解及自身水平的高低有很大的关系，而这不是 ISO 9001 标准本身所决定的。ISO 9001 标准只是质量管理体系的最低可接受准则，不能说已满足 CMM 的大部分要求。

③ 取得 CMM 第 2 级(或第 3 级)不能笼统地认为可以满足 ISO 9001 的要求。

3.2 经典软件生存周期模型

软件(开发) 生存周期是软件从需求确定、设计、开发、测试直至投入使用，并在使用中不断地修改、增补和完善，直至被新的系统所替代而停止该软件的使用的全过程。软件生存周期模型(也称软件生命周期模型)是在跨越整个软件生存周期时，实施相关的过程、活动、任务时所采用的特定结构性框架。

将早期出现的一些典型的软件生存周期模型称为经典软件生存周期模型，如瀑布模型(Waterfall Model)、原型模型(Prototype Model) 、增量模型(Incremental Model)、演化模型(Evolutionary Model)、螺旋模型(Spiral Model)、喷泉模型(Fountain Model)等。

3.2.1 瀑布模型

瀑布模型也称为线性顺序过程模型，是由温斯顿·罗伊斯(Winston Royce)于 1970 年提出的，瀑布模型将软件生存周期划分为制订计划、需求分析、软件设计、程序编码、软件测试和运行维护等 6 个阶段，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。读者可以通过图 3-2 形象地了解瀑布模型的结构。

瀑布模型是最早出现的软件开发模型，它提供了软件开发的基本框架，直到 20 世纪 80 年代早期，它一直是被广泛采用的软件开发模型，因此在软件工程中占有重要的地位。瀑布模型的每一个阶段都可以定义明确的产出物和验证准则，每一个阶段的任务完成后都可以组织相关的评审和验证，只有在评审通过后才能够开始下一个阶段的活动。

瀑布模型为软件开发和维护提供了一种有效的管理模式，它在降低软件的复杂度、促进

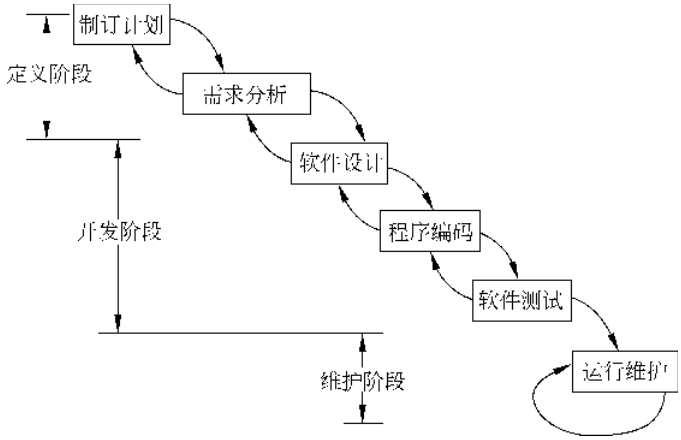

图 3-2　瀑布模型示意图

软件开发工程化方面起着显著作用。它以项目的阶段评审和文档控制为手段来有效指导软件开发过程，从而保证软件产品及时交付并达到预期的质量要求。

随着软件开发复杂性的提高，瀑布模型在大量的软件开发实践中也逐渐暴露出它的缺点：其模型结构缺乏灵活性，无法适应需求不明确或需求变化较快的软件项目开发。这些问题的存在会对软件开发带来严重影响，特别是客户要等到开发周期的晚期才能看到程序运行的测试版本，这可能导致开发出的软件并不是用户真正需要的软件，从而为软件开发人员或客户带来不必要的损失。

3.2.2　原型模型

原型模型也称为快速原型模型，它是在开发真实系统之前，构造一个原型，在该原型的基础上，逐渐完成整个系统的开发工作。在软件开发中，原型是软件的一个早期可运行的版本，它反映最终系统的部分重要特性。如果在获得一组基本需求说明后，通过快速分析构造出一个小型的软件系统，满足用户的基本要求，使得用户可在试用原型系统的过程中得到亲身感受和受到启发，做出反应和评价，然后开发者根据用户的意见对原型加以改进。随着不断试验、纠错、使用、评价和修改，获得新的原型版本，如此周而复始，逐步减少分析和通信中的误解，弥补不足之处，进一步确定各种需求细节，适应需求的变更，从而提高了最终产品的质量。

由于运用原型的目的和方式不同，原型可分为以下两种不同的类型：①废弃型，它先构造一个功能简单而且质量要求不高的模型系统，针对这个模型系统反复进行分析修改，形成比较好的设计思想，据此设计出更加完整、准确、一致、可靠的最终系统。系统构造完成后，原来的模型系统就被废弃不用；②追加型或演化型，它先构造一个功能简单而且质量要求

不高的模型系统，作为最终系统的核心，然后通过不断地扩充修改，逐步追加新要求，最后发展成为最终系统。1992 年，Andriole 给出了 6 个问题，用来帮助选择原型类型，表 3-2 指明了对这些问题的典型答案和对使用原型法的建议。

表 3-2 两种原型法的使用条件参考

问　　题	废弃型原型法	演化型原型法	其他预备工作
目标系统要解决的问题弄清楚了吗	是	是	否
问题可以被建模吗	是	是	否
客户能够确定基本需求吗	是/否	是/否	否
需求已经被建立而且比较稳定了吗	否	是	是
有模糊不清的需求吗	是	否	是
需求中有矛盾吗	是	否	是

原型的开发和使用过程叫做原型生存期。图 3-3(a)是原型生存期的模型，图(b)是模型的细化过程。

(1) 快速分析。在分析者和用户的紧密配合下，快速确定软件系统的基本要求。

(2) 构造原型。在快速分析基础上，根据基本需求，尽快实现一个可运行的系统。

(3) 运行和评价原型。用户在开发者指导下试用原型，在试用的过程中考核评价原型的特性，分析其运行结果是否满足规格说明的要求，以及规格说明描述是否满足用户愿望。

(4) 修正和改进。根据修改意见进行修改。如果用修改原型的过程代替快速分析，就形成了原型开发的迭代过程。开发者和用户在一次次的迭代过程中不断将原型完善，以接近系统的最终要求。

(5) 判定原型完成。经过修改或改进的原型，达到参与者一致认可，则原型开发的迭代过程可以结束。为此，应判断有关应用的实质是否已经掌握，迭代周期是否可以结束等。判定的结果有两个不同的转向，一是继续迭代验证，一是进行详细说明。

(6) 判断原型细部是否说明。判断组成原型的细部是否需要严格地加以说明。原型化方法允许对系统必要成分或不能通过模型进行说明的成分进行严格的详细的说明。

(7) 原型细部的说明。对于那些不能通过原型说明的所有项目，仍需通过文档加以说明。严格说明的成分要作为原型化方法的模型编入词典。

(8) 判定原型效果。考察用户新加入的需求信息和细部说明信息，看其对模型效果有什么影响，是否会影响模块的有效性。如果模型效果受到影响，甚至导致模型失效，则要进行修正和改进。

(9) 整理原型和提供文档。当项目开始前其需求不明确，或者需要减少项目的不确定性的时候，可以采用原型方法。原型模型的优点在于：①如果客户和开发者达成一致协议，原型被建造仅为了定义需求，之后就被抛弃或者部分抛弃，那么这种模型就很合适了；②吸引客户抢占市场，这是一个首选的模型。原型模型的缺点在于：①没有考虑软件的整体质

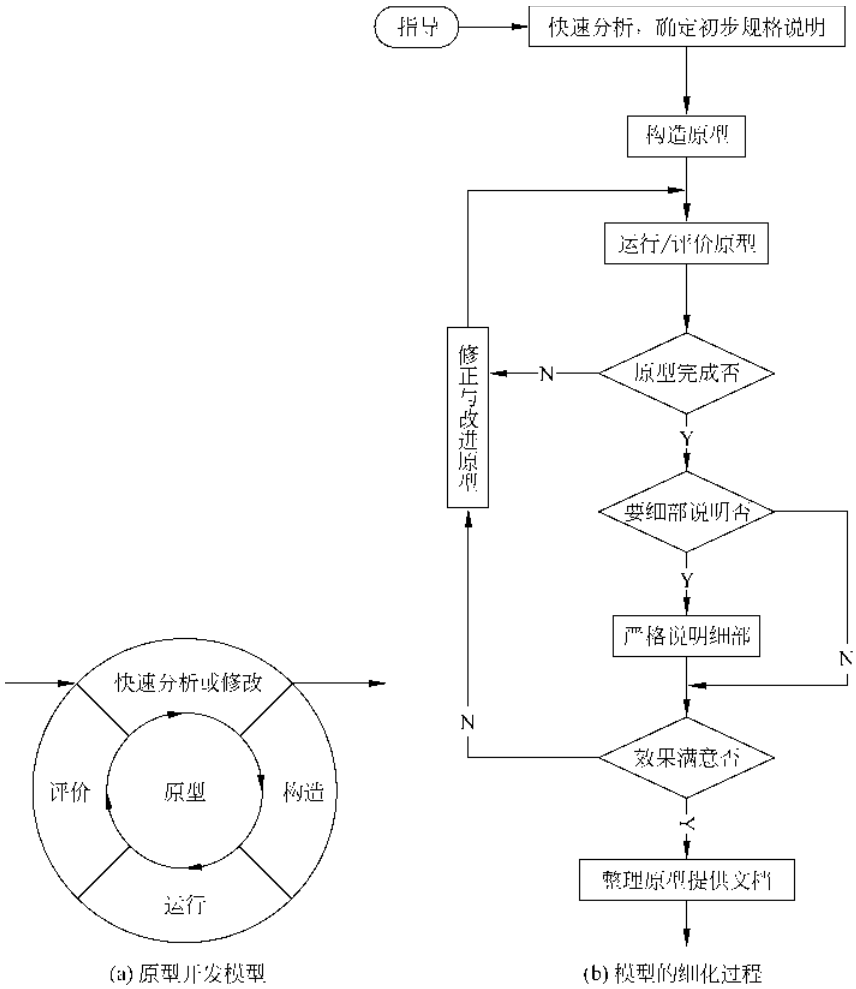

图 3-3　原型模型示意图

量和长期的可维护性；②大部分情况是不合适的操作算法被采用，目的为了演示功能，不合适的开发工具被采用仅仅为了它的方便，还有不合适的操作系统被选择等；③由于达不到质量要求产品可能被抛弃，而采用新的模型重新设计。

3.2.3　增量模型

增量模型也称为渐增模型，其结构如图 3-4 所示。使用增量模型开发软件时，把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。在使用增量模型时，第一个增量往往是实现基本需求的核心

产品。核心产品交付用户使用后，经过评价形成下一个增量的开发计划，它包括对核心产品的修改和一些新功能的发布。这个过程在每个增量发布后不断重复，直到产生最终的完善产品。例如，使用增量模型开发字处理软件，可以考虑第 1 个增量发布基本的文件管理、编辑和文档生成功能；第 2 个增量发布更加完善的编辑和文档生成功能；第 3 个增量实现拼写和语法检查功能；第 4 个增量完成高级的页面布局功能。

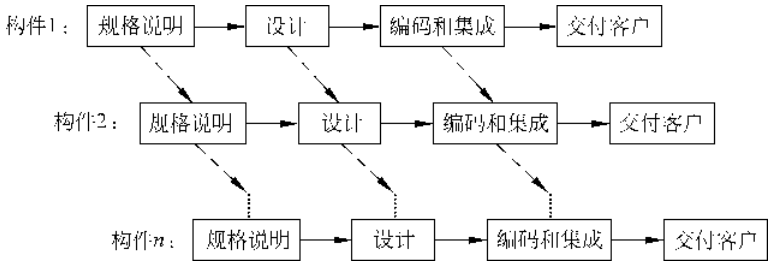

图 3-4 增量模型示意图

把软件产品分解成增量构件时，应该使构件的规模适中，规模过大或过小都不好。最佳分解方法因软件产品特点和开发人员的习惯而异。分解时唯一必须遵守的约束条件是，当把新构件集成到现有软件中时，所形成的产品必须是可测试的。

(1) 增量模型的优点。

① 整个软件产品被分解成许多个增量构件，开发人员一个构件接一个构件地向用户提交产品。从第一个构件交付之日起，用户就能做一些评估和反馈工作，显然，能在较短时间内向用户提交可完成部分工作的产品，是增量模型的一个优点；

② 逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

(2) 增量模型的缺点。

① 由于各个构件是逐渐并入已有的软件体系结构中的，所以加入构件必须不破坏已构造好的系统部分，这需要软件具备开放式的体系结构，因此增加了系统的设计难度；

② 在开发过程中，需求的变化是不可避免的。增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而使软件过程的控制失去整体性。

虽然将增量模型要求开放的软件体系结构作为其缺点，但是，从长远观点看，具有开放结构的软件拥有真正的优势，这样的软件的可维护性明显好于封闭结构的软件。因此，尽管采用增量模型比采用瀑布模型和快速原型模型需要更精心的设计，但在设计阶段多付出的劳动将在维护阶段获得回报。如果一个设计非常灵活而且足够开放，足以支持增量模型，那么，这样的设计将允许在不破坏产品的情况下进行维护。事实上，使用增量模型时开发软件和扩充软件功能(完善性维护)并没有本质区别，都是向现有产品中加入新构件的过程。