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。这套管理技术用于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，阶梯共有五级，如图 3-1 所示。

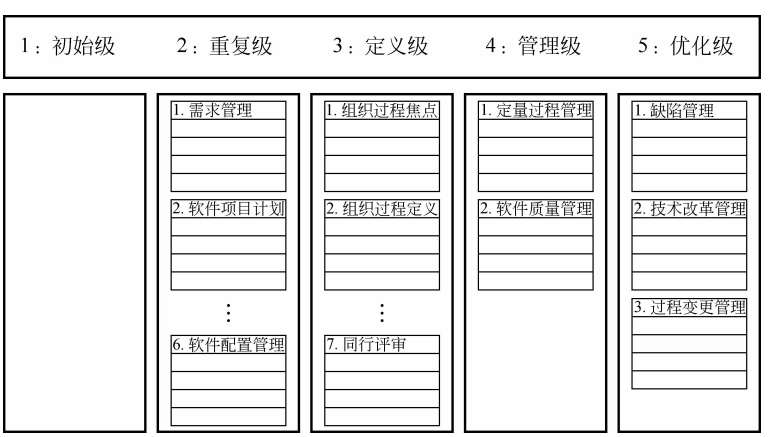


图 3-1　SW-CMM 示意图

（1）初始级。初始级的软件过程是未加定义的随意过程，项目的执行是随意的甚至是混乱的。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，那么软件企业的过程能力仍然被视为初始级。

（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，第二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。第二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五个方面。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。在第二级仅定义了管理的基本过程,而没有定义执行的步骤标准。在第三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。第四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。第五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六个公共属性进行组织,即每一个 KPA 都包含六类 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。企业必须采取相关措施,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。执行过程描述了执行 KPA 所需要的必要角色和步骤。在六个公共属性中,执行活动是唯一与项目执行相关的属性,其余五个属性则涉及企业 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,这样就导致出现了重复的培训、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 版本包括四个领域：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，也是企业在项目实施上的综合实力。在这里我们简要介绍一下 CMMI 的五个台阶。

(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，并与上级单位一起对项目与流程进行审查。企业在管理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完成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。

由上述的五个台阶我们可以看出，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，路要一步一步地走。一般地讲，应该先从管理级入手。在管理上下工夫，争取最终实现 CMMI 的第五级。

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 等。

ISO 9000 的作用如下：(1)强化品质管理，提高企业效益；增强客户信心，扩大市场份额；(2)获得国际贸易“通行证”，消除国际贸易壁垒；(3)节省了第二方审核的精力和费用；(4)在产品品质竞争中永远立于不败之地；(5)有效地避免产品责任；(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，主要是出于以下两点考虑：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我们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,也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最大的不同点有以下两个。

(1) CMMISM-SE/SW/IPPD/SS 1.1版本有四个集成成分。

(2) SW-CMM二级共有六个关键过程区域(KPA),在CMMI中增加了一个:"度量和分析"。原来的六个关键过程区域的名称和内容在CMMI中做了部分改进,但是主体内容没有大幅调整。SW-CMM四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五级共有三个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三级与CMMI三级之间,原有的七个KPA变成了十四个,CMMI对工程活动要求的KPA——原SW-CMM3"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,比如"决策分析和解决方案"、"集成化群组"等。具体如表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
二	需求管理 软件项目规划 软件项目追踪和监控 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"
三	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"、"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中的"软件集成管理",在CMMI中被分解为四个过程域——"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中的"软件产品工程",在CMMI中被分解为五个过程域——"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

续表

级别	SW-CMM 过程域	CMMI 过程域	说明与比较
四	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 的二级中，发生了较大变动
五	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，组织的变革首先上是管理文化的变革、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，不应该放在第五级，所以 CMMI 做了修正

到底是选择 SW-CMM 还是 CMMI，主要基于以下几个方面进行考虑：(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 可以平滑的转换。

ISO 9001 与 CMM 的关系如下所示。(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。两者的最大相似之处在于两者都强调“该说的要说到，说到的要做到”。CMM 强调持续改进，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，取得 CMM 等级证书，对于取得 ISO 9001 认证也是有帮助的。(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 第二级(或第三级)不能笼统地认为可以满足 ISO 9001 的要求。

3.2　传统软件开发生命周期模型

软件生命周期表明软件从需求确定、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。我们可将它划分为以下几个子阶段：(1)可行性研究；(2)需求分析和定义；(3)总体设计；(4)详细设计；(5)编码(实现)；(6)软件测试、运行和维护。据此相继产生了瀑布模型(见图 3-2)、原型模型、增量模型、进化模型、螺旋模型等。本节分别对这几种传统的软件开发生命周期模型予以介绍。

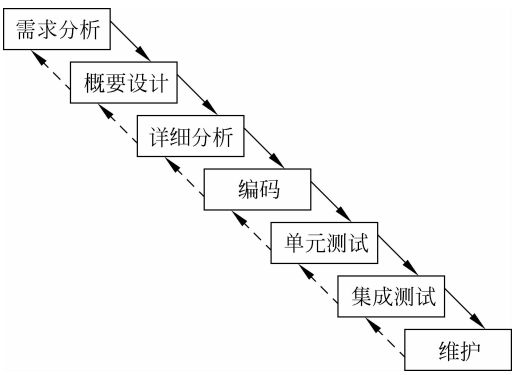


图 3-2 瀑布模型示意图

3.2.1 瀑布模型

瀑布模型(Waterfall Model),有时也称为传统生存周期模型或线性顺序过程模型,由W·Royce于1970年提出,该模型由于酷似瀑布而闻名。瀑布模型要求软件开发严格按照需求→分析→设计→编码→测试的阶段进行,如同瀑布流水,逐级下落。每一个阶段都可以定义明确的产出物和验证准则,瀑布模型在每一个阶段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进入到下一个阶段。这个模型没有反馈,一个阶段完成后,一般就不返回了,尽管实际的项目中要经常返回上一阶段。通过图3-2读者可以形象地了解瀑布模型的结构。

这种模型的线性过程太理想化,要应用到实际的软件开发过程中并想取得较好的效果是很有难度的。瀑布模型的优点在于:(1)它提供了一个模板,这个模板使得分析、设计、编码、测试和支持的方法可以在该模板下有一个共同的指导标准;(2)虽然有不少缺陷但比在软件开发中随意的状态要好得多。瀑布模型的缺点在于:(1)在大部分情况下,实际的项目难以按照该模型给出的顺序进行,而且这种模型的迭代是间接的,这很容易由微小的变化而造成大的混乱;(2)在一般情况下,客户难以表达真正的需求,而这种模型却要求如此,这种模型是不欢迎具有二义性问题存在的;(3)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,而在这时发现大的错误时,可能会引起客户的惊慌,而后果也可能是灾难性的;(4)采用这种线性模型,会经常在过程的开始和结束时碰到等待其他成员完成其所依赖的任务才能进行下去的情况,有可能花在等待的时间比开发的时间要长,我们称之为“堵塞状态”。

3.2.2 原型模型

通常原型是指模拟某种产品的原始模型。在软件开发中,原型是软件的一个早期可运行的版本,它将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

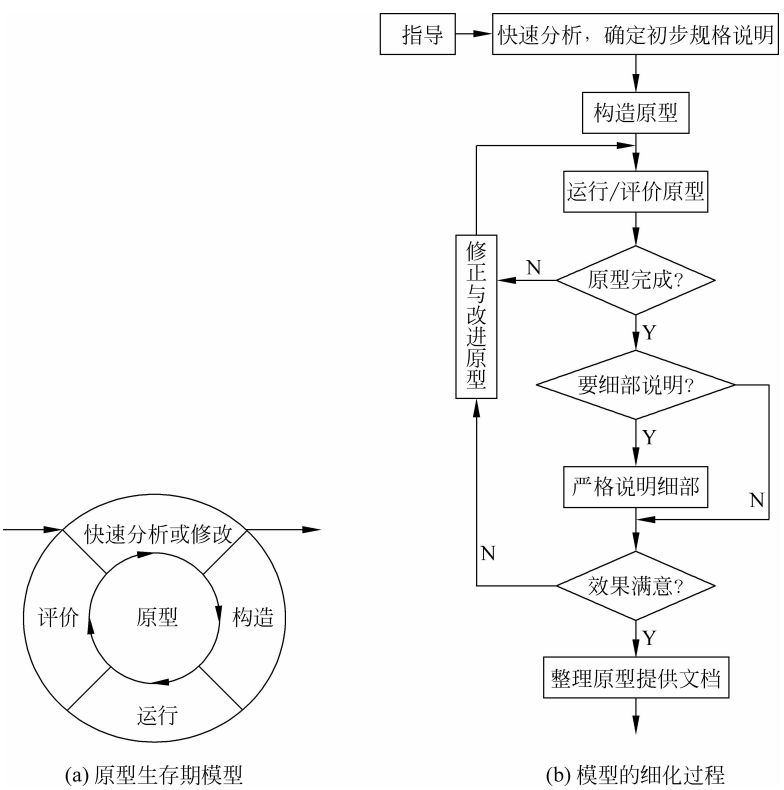
了最终产品的质量。

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。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。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。表 3-2 是对使用哪种原型方法的建议。

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 3-3(b)是模型的细化过程。



(a) 原型生存期模型　　(b) 模型的细化过程

图 3-3　原型模型示意图

对图3-3的解释如下所示。

(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) 判断原型完成。经过修改或改进的原型,若得到参与者的一致认可,则原型开发的迭代过程可以结束。为此,应判断是否已经掌握有关应用的实质,迭代周期是否可以结束等。判断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) 判断原型是否需要细部说明。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) 判断原型效果。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。

原型模型的优点有两个。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。原型模型的缺点有三个。①没有考虑软件的整体质量和长期的可维护性;②大部分情况下采用了不合适的操作算法,目的是为了演示功能,采用了不合适的开发工具,仅仅因为它很方便,还选择了不合适的操作系统等;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。

当在项目开始前,项目的需求不明确,或者需要减少项目的不确定性的时候,可以采用原型方法。类似的项目包括:需要明确系统的界面,验证一些技术的可行性等。

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,第一个增量发布基本的文件管理、编辑和文档生成功能,第二个增量发布更加完善的编辑和文档生成功能,第三个增量实现拼写和语法检查功能,第四个增量完成高级的页面布局功能。

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

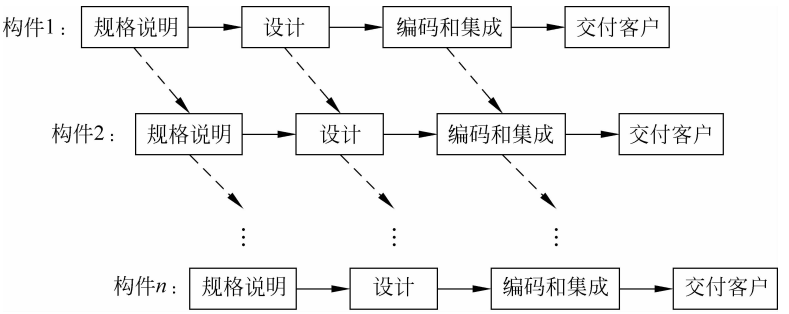


图 3-4　增量模型示意图

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。增量模型的优点有以下两个。(1)采用瀑布模型或快速原型模型开发软件时，目标都是一次就把一个满足所有需求的产品提交给用户。增量模型则与之相反，它分批地逐步向用户提交产品，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，用户就能做一些有用的工作。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。增量模型的缺点也有以下两个。(1)由于各个构件是逐渐集成到现有软件体系结构中的，所以加入构件必须不破坏已构造好的系统部分。此外，必须把软件的体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件的过程必须简单、方便，也就是说，软件体系结构必须是开放的。(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。

虽然我们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，在使用增量模型时，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。

比较适合增量模型的项目类型包括：项目开始时明确了大部分的需求，但是需求可能会发生变化的项目；对于市场和用户的把握不是很准，需要逐步了解的项目；对于有庞大和复杂功能的系统进行功能改进时需要一步一步实施的项目。

3.2.4　进化模型

该模型主要针对事先不能完整定义需求的软件开发。用户可以给出待开发系统的核心需求，并且当看到核心需求实现后，能够有效地提出反馈，以支持系统的最终设计和实现。软件开发人员根据用户的需求，首先开发核心系统。当该核心系统投入运行后，用户开始试用，完成他们的工作，并提出精化系统、增强系统能力的需求。软件开发人员根据用户的反馈，实施开发的迭代过程。每一迭代过程均由需求、设计、编码、测试、集成等阶段组成，为整

个系统增加一个可定义的、可管理的子集,如图 3-5 所示。

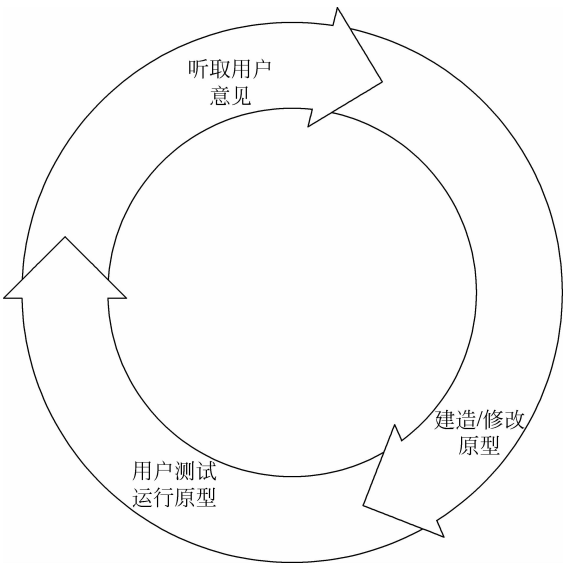


图 3-5 进化模型示意图

进化模型可看作是重复执行的多个瀑布模型。进化模型要求开发人员有能力把项目的产品需求分解为不同组,以便分批循环开发。这种分组并不是绝对随意性的,而是要根据功能的重要性及对总体设计的基础结构的影响而做出判断。有经验指出,每个开发循环以 6 周~8 周为适当的长度。增量模型与进化模型的相同点是:基本思想都是非整体开发,以渐增方式开发系统。它们的目的基本相同:使用户尽早得到部分软件,这样能听取用户反馈。两者的不同点是:增量模型在需求设计阶段是整体进行的,在编码测试阶段是渐增进行的。演化模型全部系统是增量开发,增量提交。

进化模型的优点有以下九个。(1)任何功能一经开发就能进入测试以便验证是否符合产品需求。(2)帮助导引出高质量的产品要求。如果不可能在一开始就弄清楚所有的产品需求,则可以分批取得它们。而对于已提出的产品需求,则可根据对现阶段原型的试用而做出修改。(3)风险管理可以在早期就获得项目进程数据,可据此对后续的开发循环做出比较切实的估算。提供机会去采取早期预防措施,增加项目成功的几率。(4)很有助于早期建立产品开发的配置管理、产品构建(Build)、自动化测试、缺陷跟踪、文档管理,均衡整个开发过程的负荷。(5)开发中的经验教训能反馈应用于本产品的下一个循环过程,大大提高产品质量与工作效率。(6)如果风险管理发现资金或时间已超出可承受的程度,则可以决定调整后续的开发,或在一个适当的时刻结束开发,但仍然有一个具有部分功能的、可工作的产品。(7)在心理上,开发人员早日见到产品的雏形,是一种鼓舞。(8)使用户可以在新的一批功能开发测试后,立即参加验证,以便提供非常有价值的反馈。(9)可使销售工作有可能提前进行,因为可以在产品开发的中后期取得包含了主要功能的产品原型去向客户做展示和试用。进化模型的缺点有以下四个。(1)如果所有的产品需求在一开始并不完全弄清楚的话,会给总体设计带来困难并削弱产品设计的完整性,并因而影响产品性能的优化及产品的可维护性。(2)如果缺乏严格的过程管理的话,这个生命周期模型很可能退化为一种原始的、无计

划的“试-错-改”模式。(3)可能导致心理上产生一种不求最大努力的想法,认为虽然不能完成全部功能,但还是创造出了一个有部分功能的产品。(4)如果不加控制地让用户接触开发中尚未测试稳定的功能,可能对开发人员及用户都产生负面的影响。

当需求和设计不能被准确地定义、良好地理解,或是项目引入了新的或未经证明的技术方法,或者系统功能需要向用户演示以便于演进,又或是有多种用户组,可能发生需求冲突时,则可以使用进化模型。

3.2.5　螺旋模型

螺旋模型,于 1998 年由美国 TRW 公司(B・W・Boehm)提出,是一个演化软件过程模型,它将原型模型的迭代特征与线性顺序模型中控制和系统化方面结合起来,使得软件增量版本的快速开发成为可能。它不仅体现了两个模型的优点,而且还强调了其他模型均忽略了的风险分析。在螺旋模型中,软件开发是一系列的增量发布。在早期的迭代中,发布的增量可能是一个纸上的模型或原型;在以后的迭代中,将逐步产生更加完善的被开发系统版本。

螺旋模型被划分为若干框架活动,也称为任务区域。一般情况下,有 3～6 个任务区域。图 3-6 形象地描述了包含 4 个任务区域的螺旋模型。

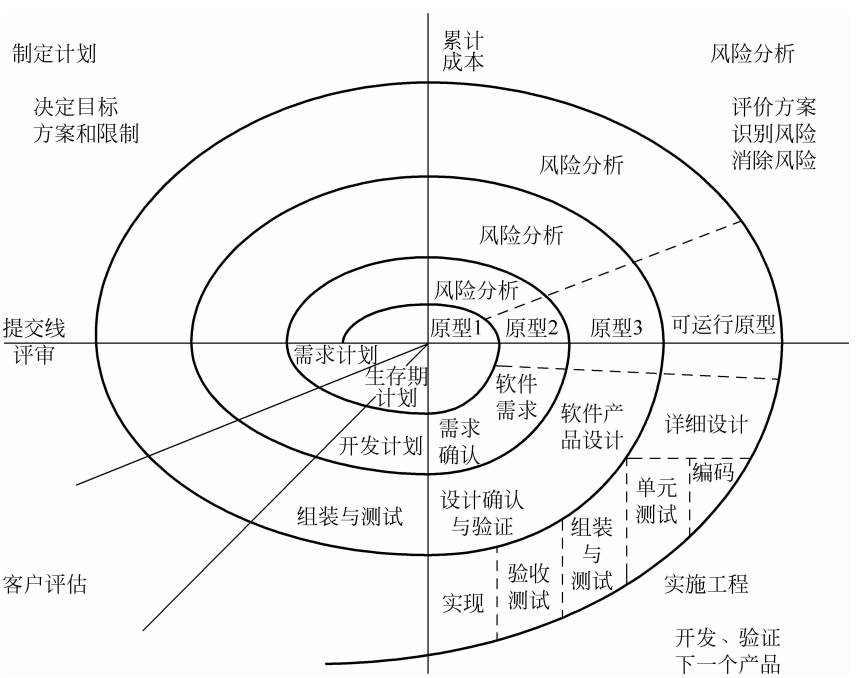


图 3-6　螺旋模型示意图

(1) 目标、选择和限制。系统要达到目标,同时要受预算、时间等条件的限制,而且必须做出一定的选择和取舍。

(2) 风险评估。基于上述目标,评估技术及管理的风险,以决定如何实施项目。

(3) 开发和测试。包括系统需求分析、概要设计、详细设计、编程、单元测试、系统测试、

验证测试等项目具体实施的各种任务。

(4) 计划。定义资源、进度及其他相关项目信息所需要完成的任务,以调整项目的目标和改善系统实施的效率。

随着演化过程的开始,软件工程项目组按顺时针方向沿螺旋移动,从核心开始。螺旋的第一圈可能产生产品的规格说明;接续的螺旋可能用于开发一个原型;随后可能是更加完善的下一个版本软件。经过计划区域的每一圈都是基于从用户评估得到的反馈,来调整费用和进度。此外,项目管理者可以调整完成软件所需计划的迭代次数。

与传统的过程模型不同,在螺旋模型中,软件交付了并不意味着结束,其适用于计算机软件的整个生命周期。一个"概念开发项目"从螺旋的核心(水平轴)开始一直持续到概念开发结束。如果概念被开发成真正的产品,过程从水平轴一个新的起点开始,意味着一个新的开发项目启动了。

本质上,具有上述特征的螺旋是一直运转的,直到软件退役。有时这个过程处于睡眠状态,但任何时候出现了改变,过程都会从合适的入口点(如产品增强)开始。

对于大型系统及软件的开发来说,螺旋模型是一个很现实的方法。因为软件随着过程的进展演化,开发者和用户能够更好地理解和对待每一个演化级别上的风险。螺旋模型使用原型作为降低风险的机制,更重要的是,它使开发者在产品演化的任一阶段均可应用原型方法。它保持了传统生命周期模型中系统的、阶段性的方法,但将其并进了迭代框架,更加真实地反映了现实世界。螺旋模型要求在项目的所有阶段直接考虑技术风险,如果应用得当,能够在风险变成问题之前降低它的危害。

螺旋模型的优点在于:(1)强调严格的全过程风险管理;(2)强调各开发阶段的质量;(3)提供机会检讨项目是否有价值继续下去。螺旋模型的缺点在于:(1)它可能难以使用户(尤其在有合同约束的情况下)相信演化方法是可控的;(2)它需要相应的风险评估的专业技术,且其成功依赖于这种专业技术,如果一个大的风险未被发现和管理,毫无疑问就会出现问题。

比较适合螺旋模型的项目类型包括:风险是主要的制约因素的项目、不确定因素和风险限制了项目进度的项目、用户对自己的需求不是很明确的项目、需要对一些基本的概念进行验证的项目、可能发生一些重大变更的项目、规模很大的项目、采用了新技术的项目等。

3.3 扩展软件开发生命周期模型

随着软件产业的蓬勃发展,对软件开发过程模型的研究也在持续进行中,除了上文提及的几种传统软件开发生命周期模型,还涌现了很多其他的开发过程模型。本节将介绍几种在业界得到广泛推广和认可的软件开发生命周期模型。

3.3.1 极限模型

2001年,为了避免许多公司的软件团队陷入不断增长的过程泥潭,一批业界专家一起概括出了一些敏捷开发过程的方法:SCRUM、Crystal、特征驱动软件开发(Feature Driven Development,FDD)、自适应软件开发(Adaptive Software Development,ASD),以及最重要

的极限编程(Extreme Programming,XP)。极限编程是于 1998 年由 Smalltalk 社群中的大师级人物 Kent Beck 首先倡导的,它是一种轻量级的开发方法,强调适应性而非预测性、强调以人为中心而不以流程为中心,以及对变化的适应和对人性的关注,其特点是轻载、基于时间、Just Enough(不多不少)、并行并基于构件的软件过程。

极限编程规定了一组核心价值和方法,消除了大多数重量型过程的不必要产物,建立了一个渐进型开发过程。该方法将开发阶段的四个活动(分析、设计、编码和测试)混合在一起,在全过程中采用迭代增量开发、反馈修正和反复测试。它把软件开发生命周期划分为用户故事、体系结构、发布计划、交互、接受测试和小型发布六个阶段,“用户故事”代替传统模型中的需求分析,由用户用自己领域中的词汇并且不考虑任何技术细节,准确地表达自己的需求。采用这种开发模型的软件过程如图 3-7 所示。

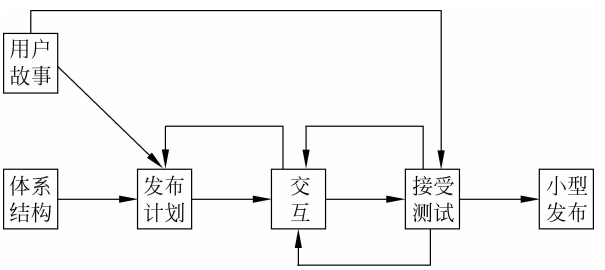


图 3-7　XP 软件开发模型示意图

XP 模型通过对传统软件开发的标准方法进行重新审视,提出了由一组规则组成的一些简便易行的过程。由于这些规则是通过在实践中观察使软件开发高效或缓慢的因素而得出的,因此它既考虑了保持开发人员的活力和创造性,又考虑了开发过程的组织性、重点性和持续性。这些实践规则列举如下。

(1) 完整团队。XP 项目的所有参与者(开发人员、客户、测试人员等)一起工作在一个开放的场所中,他们是同一个团队的成员。

(2) 计划游戏。计划是持续的、循序渐进的。每两周,开发人员就为下两周估算候选特性的成本,而客户则根据成本和商务价值来选择要实现的特性。

(3) 客户测试。作为选择每个所期望的特性的一部分,客户可以根据脚本语言以定义出自动验收测试来表明该特性可以工作。

(4) 简单设计。团队设计工作目标保持与当前的系统功能相匹配。它通过了所有的测试,不包含任何重复,表达出了编写者想表达的所有东西,并且包含尽可能少的代码。

(5) 结对编程。所有的产品软件都是由两个程序员、并排坐在一起在同一台计算机上构建的。

(6) 测试驱动开发。在程序员被分配任务后,首先要制定出该任务的测试用例,实现该任务的标志是能确保全部测试用例正确工作。各任务所使用的 TDD(Test-Driven Development)测试用例将被保留下来并应用到所有进一步的集成测试中。

(7) 改进设计。随时利用重构方法改进已经“腐化”的代码,保持代码尽可能的干净(Clean)、具有表达力。

(8) 持续集成。团队总是使系统完整地被集成。一个人签入(Check in)后,其他人员负

责代码集成。

(9) 集体代码所有权。任何结对的程序员都可以在任何时候改进任何代码。没有程序员对任何一个特定的模块或技术单独负责,每个人都可以参与任何其他方面的开发。

(10) 编码标准。系统中所有的代码看起来就好像是由单独一人编写的。

(11) 隐喻。将整个系统联系在一起的全局视图,它是系统的未来影像,是它使得所有单独模块的位置和外观变得明显直观。如果模块的外观与整个隐喻不符,那么你就会知道该模块是错误的。

(12) 可持续的速度。团队只有持久才有获胜的希望。他们以能够长期维持的速度努力工作,他们保存精力,没有一个人能够连续两周超时工作,他们把项目看作是马拉松长跑,而不是全速短跑。

XP开发模型与传统模型相比具有很大的不同,它不太强调分析和设计,在生命周期中编码活动开始得较早,因为人们认为运行的软件比详细的文档更重要。其核心思想是交流(Communication)、简单(Simplicity)、反馈(Feedback)和进取(Aggressiveness)。该模型强调小组内成员之间要经常进行交流,在尽量保证质量的前提下力求过程和代码的简单化;来自客户、开发人员和最终用户的具体反馈意见可以提供更多的机会来调整设计,保证把握正确的开发方向;进取则包含于上述三个原则中。极限编程是否是软件工程中的一个主要突破,我们很难给出一个确定的答案。对于软件开发而言,它的作用往往具有两面性,现将极限模型的优缺点总结如下。极限模型的优点在于:(1)采用简单计划策略,不需要长期计划和复杂模型,开发周期短;(2)在全过程采用迭代增量开发、反馈修正和反复测试的方法,能够适应用户经常变化的需求。极限模型的缺点在于:(1)XP目前主要是在小规模项目上应用并取得成功,但是否适用于中等规模或大规模软件产品,是需要慎重考虑的;(2)由于这个模型较新,仅在开发方面有使用该模型的少量试验数据,维护方面的数据则不多,就是说不能确定产品交付后维护成本是否降低;(3)对编码人员的经验要求高,若是在编码人员经验较少的情况下建议不要采用。

因此,与其将它作为一种软件开发生命周期模型,不如将极限编程看作是方法论,其目的是减少繁重和不必要的工件的输出、提高效率,而不是让我们过分关注阶段或过程。因此对于瀑布、增量迭代或原型,我们都可以借鉴XP方法论中的一些好的实践,这些实践都是对传统的生命周期模型的很好的补充。所以若是在项目开发过程中希望在短期内获得较高质量的代码,带来较高的工作满意度,可以考虑使用极限模型,具体情况具体分析;又或是在客户需求模糊的情况下,构建小规模软件产品,也可使用极限模型。

3.3.2 Rational统一过程

RUP(Rational Unified Process,统一软件开发过程,或统一软件过程)是一个面向对象且基于网络的程序开发方法论。根据Rational(Rational Rose和统一建模语言的开发者)的说法,RUP好像一个在线的指导者,它可以为所有方面和层次的程序开发提供指导方针、模板以及事例支持。RUP和类似的产品——例如面向对象的软件过程(OOSP),以及OPEN Process——都是理解性的软件工程工具,将把开发中面向过程的方面(例如定义的阶段、技术和实践)和其他开发的组件(例如文档、模型、手册以及代码等)整合在一个统一的框架内。

可将 RUP 软件开发生命周期看作一个二维的软件开发模型。横轴通过时间组织，是过程展开的生命周期特征，体现开发过程的动态结构。用来描述它的术语主要包括周期(Cycle)、阶段(Phase)、迭代(Iteration)和里程碑(Milestone)。纵轴以内容来组织，是自然的逻辑活动，体现开发过程的静态结构。用来描述它的术语主要包括活动(Activity)、要素(Artifact)、工作者(Worker)和工作流(Workflow)，如图 3-8 所示。

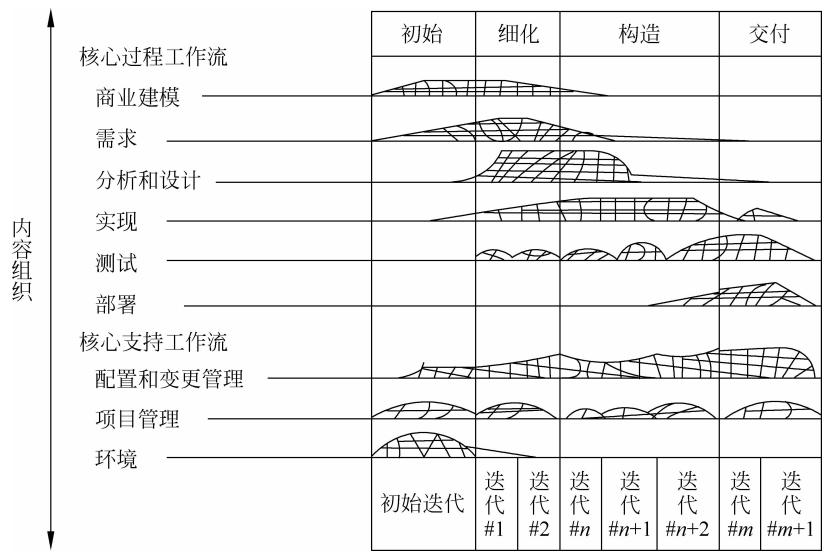


图 3-8　Rational 统一过程示意图

RUP 中的软件生命周期在时间上被分解为四个顺序的阶段，分别是：初始阶段(Inception)、细化阶段(Elaboration)、构造阶段(Construction)和交付阶段(Transition)。每个阶段结束于一个主要的里程碑(Major Milestones)；每个阶段本质上是两个里程碑之间的时间跨度。在每个阶段的结尾执行一次评估以确定这个阶段的目标是否已经满足。如果评估结果令人满意的话，可以允许项目进入下一个阶段。

(1) 初始阶段。初始阶段的目标是为系统建立商业案例并确定项目的边界。为了达到该目的必须识别所有与系统交互的外部实体，在较高层次上定义交互的特性。本阶段具有非常重要的意义，在这个阶段中所关注的是整个项目进行中的业务和需求方面的主要风险。对于建立在原有系统基础上的开发项目来讲，初始阶段可能很短。初始阶段结束时将出现第一个重要的里程碑：生命周期目标(Lifecycle Objective)里程碑。生命周期目标里程碑可评价项目基本的生存能力。

(2) 细化阶段。细化阶段的目标是分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中拥有最高风险的元素。为了达到该目的，必须在理解整个系统的基础上，对体系结构做出决策，包括其范围、主要功能和诸如性能等非功能需求。同时为项目建立支持环境，包括创建开发案例、模板、准则并准备工具。细化阶段结束时将出现第二个重要的里程碑：生命周期结构(Lifecycle Architecture)里程碑。生命周期结构里程碑为系统的结构建立了管理基准并使项目小组能够在构建阶段中进行衡量。此刻，要检验详细的系统目标和范围、结构的选择以及主要风险的解决方案。

(3) 构造阶段。在构造阶段,所有剩余的构件和应用程序功能被开发并集成为产品,所有的功能将被详细测试。从某种意义上说,构造阶段是一个制造过程,其重点放在管理资源及控制运作,以优化成本、进度和质量。构造阶段结束时将出现第三个重要的里程碑:初始功能(Initial Operational)里程碑。初始功能里程碑决定了产品是否可以在测试环境中进行部署。此刻,要确定软件、环境、用户是否可以开始系统的运作。此时的产品版本也常被称为 beta 版。

(4) 交付阶段。交付阶段的重点是确保软件对最终用户是可用的。交付阶段可以跨越几次迭代,包括为发布做准备的产品测试、基于用户反馈的少量的调整。在生命周期的这一点上,用户反馈应主要集中在产品调整,设置、安装和可用性问题,所有主要的结构问题应该已经在项目生命周期的早期阶段解决了。在交付阶段的终点将出现第四个里程碑:产品发布(Product Release)里程碑。此时,要确定目标是否实现,是否应该开始另一个开发周期。在一些情况下这个里程碑可能与下一个周期的初始阶段的结束重合。

RUP 中有九个核心工作流,分为六个核心过程工作流(Core Process Workflows)和三个核心支持工作流(Core Supporting Workflows)。尽管六个核心过程工作流可能使人想起传统瀑布模型中的几个阶段,但应注意迭代过程中的阶段是完全不同的,这些工作流在整个生命周期中一次又一次被访问。九个核心工作流在项目中轮流被使用,在每一次迭代中以不同的重点和强度重复。

(1) 商业建模(Business Modeling)。商业建模工作流描述了如何为新的目标组织开发一个构想,并基于这个构想在商业用例模型和商业对象模型中定义组织的过程、角色和责任。

(2) 需求(Requirements)。需求工作流的目标是描述系统应该做什么,并使开发人员和用户就这一描述达成共识。为了达到该目标,要对需要的功能和约束进行提取、组织、文档化;最重要的是理解系统所解决问题的定义和范围。

(3) 分析和设计(Analysis & Design)。分析和设计工作流将需求转化成未来系统的设计,为系统开发一个健壮的结构并调整设计使其与实现环境相匹配,优化其性能。分析设计的结果是产生一个设计模型和一个可选的分析模型。设计模型是源代码的抽象,由设计类和一些描述组成。设计类被组织成具有良好接口的设计包(Package)和设计子系统(Subsystem),而描述则体现了类的对象如何协同工作,以实现用例的功能。设计活动以体系结构设计为中心,体系结构由若干结构视图来表达,结构视图是整个设计的抽象和简化,该视图中省略了一些细节,使重要的特点体现得更加清晰。体系结构不仅仅是良好设计模型的承载媒介,而且在系统的开发中能提高被创建模型的质量。

(4) 实现(Implementation)。实现工作流的目的包括以层次化的子系统形式定义代码的组织结构;以组件的形式(源文件、二进制文件、可执行文件)实现类和对象;将开发出的组件作为单元进行测试,以及集成由单个开发者(或小组)所产生的结果,使其成为可执行的系统。

(5) 测试(Test)。测试工作流要验证对象间的交互作用,验证软件中所有组件的正确集成,检验所有的需求已被正确的实现,识别并确认缺陷在软件部署之前被提出并得到处理。RUP 提出了迭代的方法,意味着在整个项目中进行测试,从而尽可能早地发现缺陷,从根本上降低了修改缺陷的成本。测试类似于三维模型,分别从可靠性、功能性和系统性能来

进行。

(6) 部署(Deployment)。部署工作流的目的是成功地生成版本并将软件发布给最终用户。部署工作流描述了那些与确保软件产品对最终用户具有可用性的相关的活动,包括:软件打包、生成软件本身以外的产品、安装软件、为用户提供帮助。在有些情况下,还可能包括计划和进行 beta 版测试、移植现有的软件和数据以及正式验收。

(7) 配置和变更管理(Configuration & Change Management)。配置和变更管理工作流描绘了如何在多个成员组成的项目中控制大量的产物。配置和变更管理工作流提供了准则来管理演化系统中的多个变体,跟踪软件创建过程中的版本。工作流描述了如何管理并行开发、分布式开发、如何自动化创建工程。同时也阐述了对产品修改原因、时间、人员,保持审计记录。

(8) 项目管理(Project Management)。软件项目管理用于平衡各种可能产生冲突的目标,管理风险、克服各种约束并成功交付使用户满意的产品。其目标包括:为项目的管理提供框架,为计划、人员配备、执行和监控项目提供实用的准则,为管理风险提供框架等。

(9) 环境(Environment)。环境工作流的目的是向软件开发组织提供软件开发环境,包括过程和工具。环境工作流集中于配置项目过程中所需要的活动,同样也支持开发项目规范的活动,提供了逐步的指导手册并介绍了如何在组织中实现过程。

与传统的瀑布模型相比较,迭代过程具有的优点之一是:降低了在一个增量上的开支风险。如果开发人员重复某个迭代,那么损失的只是这一个开发有误的迭代的花费;迭代过程降低了产品无法按照既定进度进入市场的风险。通过在开发早期就确定风险,可以尽早来排除风险而不至于在开发后期匆忙排除风险;迭代过程加快了整个开发工作的进度。因为开发人员清楚问题的焦点所在,他们的工作会更有效率;由于用户的需求并不能在一开始就做出完全的界定,它们通常是在后续阶段中不断细化的。因此,迭代过程这种模式使适应需求的变化会更容易些。

但 RUP 是一个通用的过程模板,包含了很多开发指南、工件、开发过程所涉及的角色说明。由于它非常庞大所以对于具体的开发机构和项目,当使用 RUP 时还要做裁剪,也就是要对 RUP 进行配置。RUP 就像一个元过程,通过对 RUP 进行裁剪可以得到很多不同的开发过程,这些软件开发过程可以看作 RUP 的具体实例。RUP 裁剪可以分为以下几步:(1)确定本项目需要哪些工作流,RUP 的九个核心工作流并不总是需要的,可以取舍;(2)确定每个工作流需要哪些要素;(3)确定四个阶段之间如何演进;(4)确定每个阶段内的迭代计划,规划 RUP 的四个阶段中每次迭代开发的内容;(5)规划工作流内部结构。

RUP 具有很多长处:提高了团队生产力,在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面,针对所有关键的开发活动为每个开发成员提供了必要的准则、模板和工具指导,并确保全体成员共享相同的知识库。它建立了简洁和清晰的过程结构,为开发过程提供较大的通用性。但同时它也存在一些不足:RUP 只是一个开发过程,并没有涵盖软件过程的全部内容;此外,它不支持多项目的开发结构,这在一定程度上降低了在开发组织内大范围实现重用的可能性。可以说 RUP 是一个非常好的开端,但并不完美,在实际的应用中可以根据需要对其进行改进并可以用 OPEN process 和 OOSP 等其他软件过程的相关内容对 RUP 进行补充和完善。

3.3.3 微软产品开发周期模型

微软是世界上最大的软件公司,但微软并没有通过CMM认证,不使用RUP,也不使用XP。该公司有自己独特的软件开发过程——微软产品开发周期模型PCM(Product Cycle Model)。微软产品开发周期模型是微软三十多年实际开发经验的精髓,微软的所有产品,从最初的产品策划到编程、beta版发行、正式版本的发布、下一个版本的开发,都遵循该周期模型。微软产品开发周期模型是整个微软开发流程的核心和基础。合理的人员配置、合理的团队架构保证了微软能够开发出符合用户需求的高质量产品。

微软公司每一个产品的发布,往往需要多个开发团队的共同努力。每个团队都有自己的想法,实施什么样的开发流程也是灵活的,但总体上是一样的,就是采用成熟的PCM流程,该软件开发周期通常分为以下四个阶段,如图3-9所示。

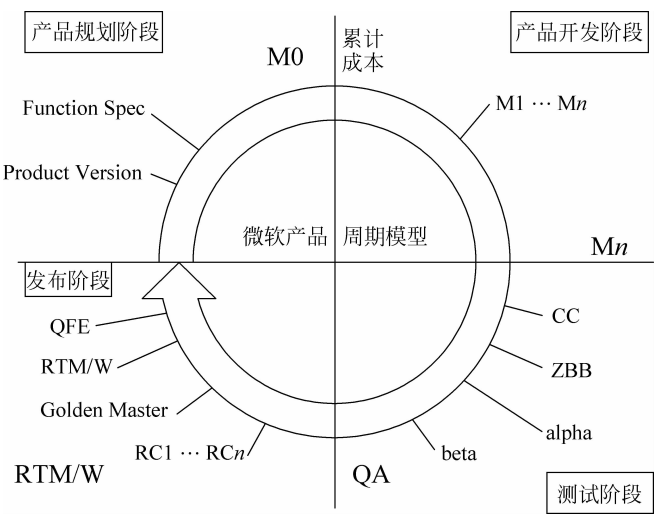


图3-9 微软产品开发模型示意图

(1) 产品规划阶段。先做市场分析和产品调查,看市场有无此软件产品的需求,以及竞争对手有哪些,这就是Product Vision的工作内容。接着就是Function Spec,主要是设计软件要实现哪些功能,解决用户什么样的问题。细致的方面包括选择什么语言来编写代码,所支持的平台以及开发所用的模型。

(2) 产品开发阶段。就是依据设计编写代码,并同时让测试人员找bug。M1…Mn表示第1个~第n个里程碑,每达到一个里程碑,就可以让测试人员同时进行工作,如进行单元测试。

(3) 测试阶段。达到所有里程碑,进入CC(Code Completion)状态,刚编写的代码肯定存在诸多问题,测试就是找出这些问题中的大部分,然后进行修正,得到一个相对稳定的软件版本。这里的ZBB是指Zero Bug Bounce,即软件运行48小时内不出现bug,然后就是alpha直至beta版本的软件面世,在较大规模地推向市场前,应检查软件的使用性能并予以修正。

(4) 发布阶段。从Release 1.0版的发布到Release N.0版,黄金版本(Golden

Masters)印制光碟提交给制造商作样本,至正式版本 RTM/W(Release To Manufacturing/Web)的发布,做好 QFE(Quick-Fix Engineering),厂商为了及时升级功能或者修改缺陷而做的改动(支持)叫做 QFE,简单说就是提供补丁的工作。

第一个阶段是产品规划阶段,首先要有一个远景规划,而且要有很清晰的长远计划。这个产品到底要达到什么样的目的,在市场上与竞争对手进行竞争的过程中要完成什么样的战略性目的。接下来在功能点规格说明部分有很多设计工作要做:设计规范书的总体功能设计、具体的功能设计、详细的使用界面的设计、开发执行和构架设计、软件的组件构架和整合的设计等。详细的功能设计,以使用方案和需求总结为基础,每个功能设计对照使用方法可证实其合理性。前面的设计做得越多,对后来工作的开展越是有利。在设计工作中很重要的是,先要了解客户的使用方案,他们怎么样使用软件,使用软件的步骤到底是怎样的;然后由此决定要怎样的功能;最后所有的功能需求决定了最终的设计。用户的使用流程要清楚,微软是非常重视这一点的,所有的开发都要经历上述三个步骤。从使用方案到功能需求,然后到功能设计。第二个阶段是产品开发阶段,前面的工作都做完了,就进行具体开发,即编写程序,不仅是编写功能的组件本身,此外还包括审核设计文档、安装并配置开发环境、代码签入工作、每日产品生成以及管理 bug 数据库等。第三个阶段就是测试阶段,程序写完了(Code Completion,CC),就会发现很多 bug,发现很多需要进行修改的缺陷,这就要展开所谓的质量保证工作。对微软而言,这个工作很简洁,用所谓的战争三国会议来决定修改工作:任何决定,都不是由一方做出,而是由三方代表(项目经理、开发者和 QA)做出。最后进入第四个阶段——发布阶段,从 Release 1.0 版本开始,直至 Release *N*.0 版本的发布,软件才能稳定,纠错、测试后发布最终产品,把发行版本刻录到 CD(DVD)上等。

总体来说,在微软产品的开发中使用了完善的团体结构,还使用了完整的流程。利用这个项目流程管理的方法可帮助流程管理,并提高软件开发的质量。管理好四个流程的执行,事先要制定出完整的衡量标准。此外要进行重复循环的运作,利用重复的四个周期来进行不断的调整,每个周期都要进行管理,依据事先定好的规划、衡量标准,一个周期接一个周期地管理。如果前面的周期已经被延迟,你知道后面的工作一定不能准时完成,那么你应该告诉你的其他团队,让他们帮你的计划做即时的调整。这些工作对于一个项目管理人员,或一个领导而言是很重要的。如果能顺利完成这些工作,团队将使用良好的流程,对软化开发的成功是一个很重要的部分。软件开发是很困难的一件事,要符合市场、客户的要求,运用这些理论和实践对于帮助我们进行管理是非常重要的。

3.4　案 例 分 析

HRMS 系统(人力资源管理系统,是为某跨国企业的 ISS 部门而开发的)的生存期模型选择过程:针对本项目的开发特点,参考企业的生存期模型说明和软件过程体系,决定采用迭代增量式模型(见图 3-10),理由如下所示。

(1) 人力资源管理系统的全部功能分成通用功能和分类业务处理功能两大类,因此可以先基于通用功能做出一个模块的最小使用版本,再逐步添加其余的功能。这样一来,用户可以在先试用最小版本的同时,提出更多明确的需求,这有助于下一阶段的开发,大大减小

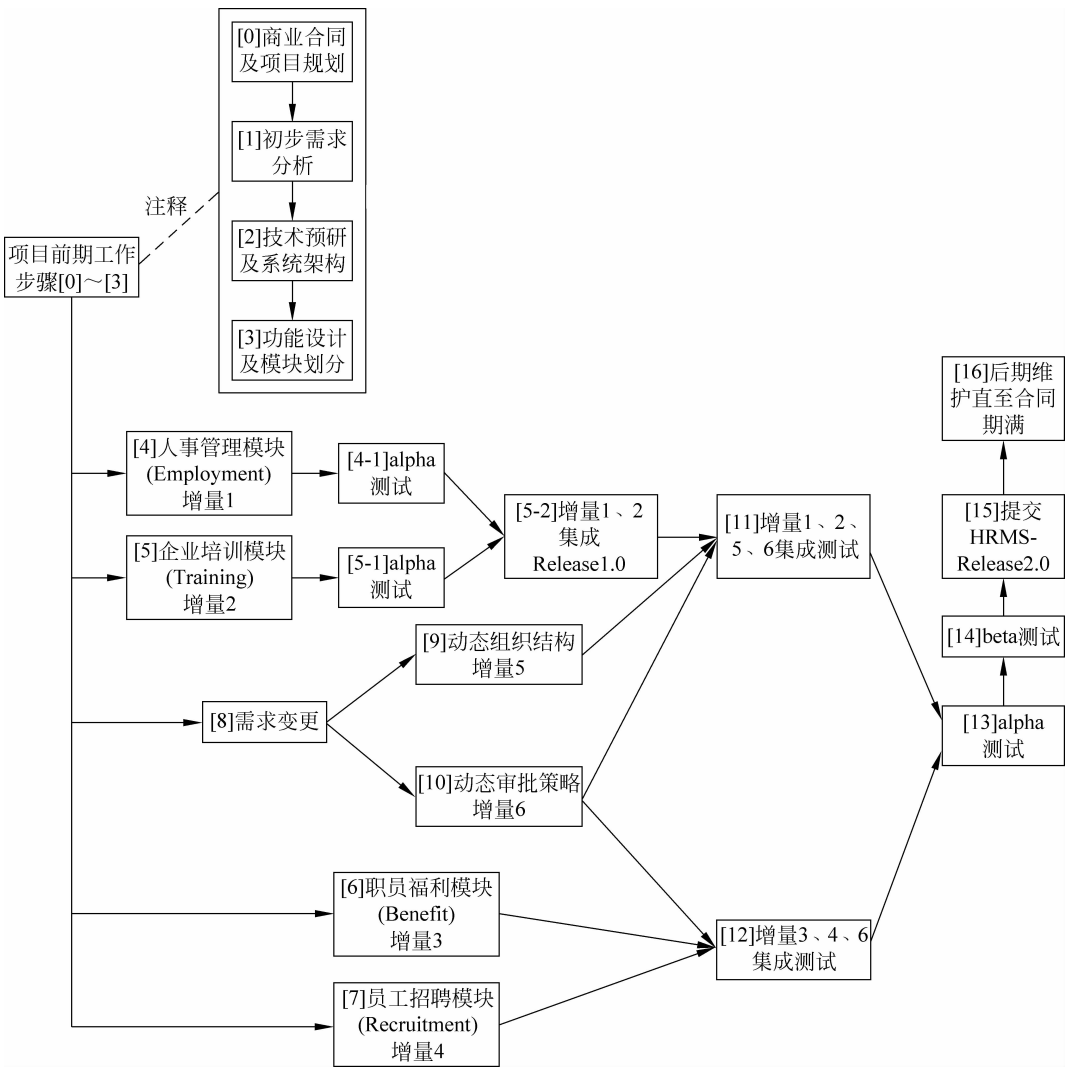


图 3-10 HRMS项目生存期模型

了开发的风险。

(2) 在人力资源管理系统需求中,要求系统有可扩充性。若使用增量式模型,可以保证系统的可扩充性。用户明确了需求的大部分,但也存在不太详尽的地方。如其中的通用查询自定义控件,就其要实现的功能只是提出了大概的意向,并没有完整确定的要求;系统参数模块只提到"应提供一个标准的、可维护的系统参数解决方案"。这样,只有等到一个可用的产品出来,才能通过客户使用这个可用产品,然后进行评估,将评估结果作为下一个增量的开发计划,下一个增量发布一些新增的功能和特性,直至产生最终完善的产品。

(3) "系统要求有可扩充性,可以在现有系统的基础上,方便迅捷地加挂其他功能模块"——也说明用户可能会增加新的需求,事实上也确实如此,在 Employment、Training 模块的开发中途,客户提出了动态组织结构和动态审批策略的新需求。

(4) 对一个管理方式已经非常成熟的公司,要其短时间内完全舍弃原有的管理方式,用

人力资源管理系统替代全部管理，这是不实际的。所以，可以从最基础的做起，逐步扩充其应用，所以可选用增量式模型来开发 HRMS 系统。

(5) 本项目具备迭代增量式模型的其他特点：

- 项目复杂程度为中等。
- 大部分需求是稳定的，拥有良好理解的，但存在某些不明确的概念。
- 产品和文档的再使用率会很高。
- 有经过证明的、成熟的设计和技术。

项目的持续期达一年半，客户需要间歇性发布。

该项目生存周期中的各阶段定义如下所示：

(1) 商业合同及项目规划。阶段目标：根据合同和前期的需求分析，确定项目的规模、时间计划和资源需求，同时确立项目的组织机构，以及组成人员需承担的相应任务，此外还有相应的规章制度要予以确定。从此阶段开始做好配置管理工作，对于所有的文档，各版本的源程序都要纳入配置库，以供追踪。

输入：合同文本和 SOW。

过程：项目规划、计划确认、团队组成、职责分配和制度审核。

输出：项目计划、配置管理计划、标准和规章制度。

(2) 初步需求分析。阶段目标：确定客户的需求。

输入：项目计划和 SOW。

过程：需求获取、需求分析、需求控制和软件需求评审。

输出：原型系统和需求规格。

(3) 技术预研及架构设计。阶段目标：根据系统需求，确定技术手段与前期的培训计划，完成 B/S 结构系统的架构设计。

输入：原型系统和需求规格。

过程：总体设计、概要设计评审和同行评审。

输出：系统设计说明书和技术培训计划。

(4) 功能设计及模块划分。阶段目标：完成详细的功能设计，归并后组织成不同的模块开发任务，以实现分时段增量开发的目的，同时进一步完善项目计划。

输入：系统设计说明书和项目计划。

过程：功能设计、详细设计评审和同行评审。

输出：功能设计说明书、数据库结构定义和产品审计要素。

(5) Employment 增量实现。阶段目标：实现系统的人事管理(Employment)功能，而后与 Training 模块合并。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码、与 Training 模块集成 HRMS-Release 1.0 版和产品审计报告。

(6) Training 增量实现。阶段目标：实现系统的培训考核(Training)功能，而后与 Employment 模块合并。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码、与 Employment 模块集成 HRMS-Release 1.0 版和产品审计报告。

(7) 需求变更。阶段目标：根据客户新需求，完成动态组织结构、动态审批策略模块的前期工作，以便在下一阶段开展编码工作。

输入：需求变更报告书。

过程：需求分析、软件需求评审、总体设计、功能设计和同行评审。

输出：需求规格、概要设计说明书、功能设计说明书、数据库结构定义和产品审计要素。

(8) 动态组织结构增量实现。段目标：实现系统的企业动态组织结构搭建(Dynamic-Organization)功能，而后分别与各模块集成。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码，与 Dynamic-Policy 模块、HRMS-Release 1.0 版本集成测试，与 Dynamic-Policy 模块、Benefit 模块、Recruitment 模块集成测试，产品审计报告。

(9) 动态审批策略增量实现。阶段目标：实现系统的企业动态审批策略构建(Dynamic-Policy)功能，而后分别与各模块集成。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码，与 Dynamic-Policy 模块、HRMS-Release 1.0 版本集成测试，与 Dynamic-Policy 模块、Benefit 模块、Recruitment 模块集成测试，产品审计报告。

(10) Benefit 增量实现。阶段目标：实现系统的福利配给管理(Benefit)功能，而后与 Recruitment 模块、Dynamic-Organization 模块、Dynamic-Policy 模块合并。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码，与 Dynamic-Organization 模块、Dynamic-Policy 模块、Recruitment 模块集成测试，产品审计报告。

(11) Recruitment 增量实现。阶段目标：实现系统的招聘管理(Recruitment)功能，而后与 Benefit 模块、Dynamic-Organization 模块、Dynamic-Policy 模块合并。

输入：功能设计说明书、数据库结构定义和产品审计要素。

过程：编码、数据库设计、代码走查、代码评审、单元测试、管理评审和 alpha 测试。

输出：源代码，与 Dynamic-Organization 模块、Dynamic-Policy 模块、Benefit 模块集成测试，产品审计报告。

(12) 系统集成阶段。阶段目标：集成所有模块的最新版本，通过产品提交前的最后测试。

输入：各模块最新版本、测试计划和测试用例。

过程：集成测试、系统测试、压力测试、功能检查、管理评审、alpha 测试和 beta 测试。

输出：系统软件包 Release 2.0、测试报告和产品说明书。

(13) 产品提交。阶段目标：产品可投入使用。

输入：系统软件包 Release 2.0。

过程：验收测试和产品提交。

输出：验收报告。

注：生存周期模型中的过程定义可以参照企业的质量保证体系并结合项目的具体特点来决定，因为公司的流程已覆盖到项目开发、管理的所有方面，包括从最开始的合同到最后软件的产品提交，都有相应的过程规定，基本上已形成一种工业化的软件开发，所以为形成一个良好的软件开发环境奠定了基础。

例如，系统设计过程及产品标准的定义如下所示。

参与角色

R1：项目经理

R2：开发经理

R3：设计人员

进入条件

E1：到达项目计划规定的系统设计时间

输入

I1：需求规格

活动

A1：设计人员了解业务需求并仔细阅读需求规格

A2：设计人员收集了解同类项目的技术框架

A3：开发经理领导设计人员通过具体的业务分析和企业成熟的技术框架进行系统设计

A4：设计人员在进行系统设计时，应按照系统设计的标准模板进行，要求如下：

- 完整、正确、如实地说明每个模块的流程和数据库表
- 用中文进行描述，并用小四号文字

A5：开发经理负责监督设计人员设计文档的同行评审

A6：开发经理主持设计正式评审，同时要求项目经理和质量经理参加

A7：设计人员根据评审结果进行修订和补充，并形成最终系统设计文档

A8：开发经理负责将系统设计过程中无法解决的问题以事件报告形式提交给项目经理，由项目经理进行跟踪解决

输出

O1：系统设计文档（格式标准见企业质量体系）

完成标志

F1：系统设计评审通过，纳入配置库

3.5　本章小结

本章讲述了软件开发过程管理需要掌握的部分知识，介绍了 ISO 9000、CMM 和 CMMI 三种常见的软件过程改进方法，并且比较了它们之间的异同，对于选取哪种方法给予了建议。

本章详细介绍了多种软件开发生命周期模型的特点和优缺点，对于软件开发中的相当重要的项目选型工作提供了参照。

此外还介绍了质量计划的定义和详细的模板。质量计划的制定对于软件质量控制的重要性非同小可,它涉及的范围很广,需要制定的内容相当多,部分内容读者可以在其他章节详细了解。

3.6 复习思考题

1. CMM 和 CMMI 的五个级别分别是什么? CMM 和 CMMI 的关系是什么?
2. 在软件企业中推行 ISO 9000 的意义是什么?
3. 传统的软件开发生命周期可以分为哪几个子阶段?
4. 原型模型可以细分为哪两种? 它们的内容是什么?
5. 你觉得进化模型和螺旋模型有哪些相似之处? 它们的核心思想是什么?
6. 质量体系、质量手册和质量计划的联系是什么?
7. 在需求分析阶段需要监控的关键元素是什么?