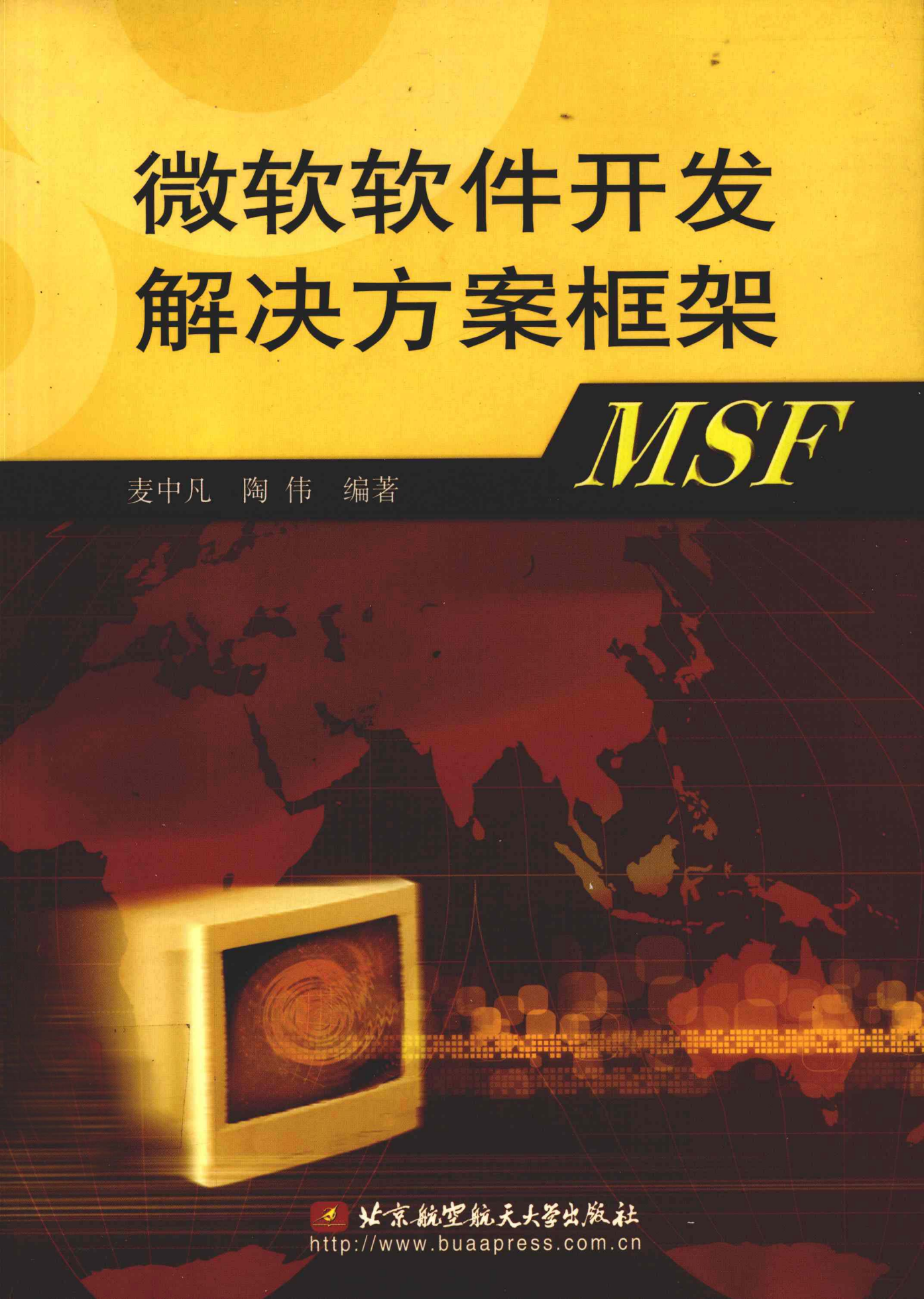
微软软件开发解决方案框架

MSF

麦中凡　陶 伟　编著



北京航空航天大学出版社
http://www.buaapress.com.cn

微软软件开发解决方案框架 MSF

麦中凡　陶　伟　编著

北京航空航天大学出版社
http://www.buaapress.com.cn

内 容 简 介

微软解决方案框架 MSF(Microsoft Solution Framework)是微软公司,以及微软的产品开发者、IT 组织、咨询专家、客户和全球范围合作伙伴的软件开发的经验的总结。MSF 是一种实用的软件工程方法。本书介绍了 MSF 的 3 个基础模型:风险管理模型、小组模型及过程模型;详述了 MSF 的 4 种软件开发范型:企业体系结构原理、应用开发原理、构件设计原理及基础设施部署原理;最后还讨论了如何采用 MSF 来提高软件过程成熟度,分析了 MSF 与 CMM 的关系,介绍了瑞理统一过程 RUP 和极限编程 XP,比较了 RUP,XP 和 MSF。本书适用于软件开发的从业人员、软件专业的高年级本科生和研究生,亦可作为软件学院研究生的教材。

图书在版编目(CIP)数据

微软软件开发解决方案框架 MSF/麦中凡等编著.
京:北京航空航天大学出版社,2003.6
ISBN 7-81077-338-0

Ⅰ.微… Ⅱ.麦… Ⅲ.软件开发 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字(2003)第 040990 号

微软软件开发
解决方案框架 MSF
麦中凡 陶 伟 编著
责任编辑 王 瑛
*
北京航空航天大学出版社出版发行
北京市海淀区学院路 37 号(邮编 100083) 发行部电话:(010)82317024 传真:(010)82328026
http://www.buaapress.com.cn
E-mail:bhpress@263.net
河北省涿州市新华印刷厂印装 各地书店经销
*
开本:787×1092 1/16 印张:23.25 字数:595 千字
2003 年 6 月第 1 版 2003 年 6 月第 1 次印刷 印数:5 000 册
ISBN 7-81077-338-0 定价:36.00 元

前　言

应用软件开发一直是人们关注的问题，谁都知道软件开发是用某种编程语言编写程序。学过软件工程的人都知道，编程仅仅是实现软件开发的一个阶段。它前有分析、设计，后有测试、交付和维护。软件开发过程是整个软件生命周期全过程的一部分。创建、使用、维护及退役是它的全过程，简称软件过程(software process)。当然，从无到有创建软件的开发过程是核心过程。

由于软件越来越大，越来越复杂，创建它所消耗的资金动辄几万、几十万，甚至几百万，并且开发完成的软件也不一定可以使用；所以，必须以工程的方法才能按时、保质、按客户的需求完成它，且不超过预算。当今正式软件的开发过程也是软件工程过程。

工程的单位是项目(project)。项目是人们按照合同要求完成的各项活动(工作)，以得到工程产物(可以是商品，如一架飞机；可以是服务，如电话系统；也可以是完整产品的一部分，如一座楼基)。合同的甲方，即出钱做此工程的业主，也叫客户(customer)。客户只负责验收工程产品，并为该项目付款，不一定是该工程产物的最终用户(end user)。例如，航空公司是业主，驾驶员和地勤人员才是飞机的使用者。显然，该产品的使用价值和工作性能只有最终用户最清楚。合同的乙方也叫承包商(contractor)，即该项目的实施者。承包商从项目立项、签定合同、项目实施到项目交付，组织人力和物力，保质、按期交付承制的产物；还要做好交付后的服务，培训最终用户的操作和维护；有了问题，还要返工、返修，甚至赔偿因本产品故障对客户方业务过程造成的损失(按合同定义的赔偿条款)。

工程项目是在市场环境下人们的社会生产活动。每一项活动都要精心核算成本，计算投入所带来的经济效益；为此，项目必须有周密的计划，并在执行过程中及时调整，因为无论多么周密的计划，也无法预料计划执行期间出现的偶发事件，

如主要干将生病、停电导致通信中断等。工程项目按期完工是第一位的，否则要受到违反合同的处罚。然而计划的拖延是工程项目的通病，直到交付期临近，才不惜工本抢进度，做高价互易（请高手，买成熟部件），这种现象时有发生。

软件产品的开发过程和一般工程产品的工程过程完全一样，只是软件产品是意识产品，投入的原材料少，投入的人力较多。每个人配备机器和必要的环境工具软件（甚至开发者可自备），就可投入生产，而开发出的产品就是源代码和文档。软件从某种意义上讲，交付、传输及更改极其容易，不像物质产品，一旦交付，缺陷陪伴终生，很难更改。

作为意识产品的软件，其开发与交付是一个复杂的过程，一些困难始终困扰着软件开发组织。虽然企业在软件开发方面投入了大量的资金和人力，然而仍然无法获得预期的系统。无论是过去，还是现在，都有许多这样失败的例子：软件无法按时交付，系统太昂贵，系统经常出故障，或者根本不满足客户的业务需求。

虽然软件开发仍然面临着诸多挑战，但是为了提高软件开发效率，降低成本，保证质量，软件开发组织仍然在不懈地努力。有的申请 ISO9001 认证，有的做 CMM（Capability Maturity Model）评估，有的提倡极限编程 XP（eXtreme Programming），有的推行 RUP（Rational Unified Process），众说纷纭，莫衷一是。当然各种做法自有它的道理，那么有没有一种简单易行、富有成效的做法呢？有！这里我们推荐微软的 MSF（Microsoft Solution Framework）。

本书试图从软件方法学和开发技术的角度介绍微软公司是如何进行软件开发的，并且与其它的软件开发方法和理论作比较，希望能对国内软件企业有所启发和借鉴。

本书在编写过程中得到微软（中国）公司的帮助。微软公司向我们提供了 MSF 的相关材料，微软公司的王静茹女士和李东方先生为我们提供了咨询。这里向微软公司以及王静茹女士和李东方先生表示感谢。

编著者
于北京航空航天大学
2002 年 12 月 16 日

目 录

第0章 绪 论

MSF(Microsoft Solution Framework)是微软公司解决方案框架的缩写，是微软咨询服务MCS(Microsoft Consulting Service)的一部分，旨在帮助客户以IT解决方案解决自己的业务问题。MSF总结了微软公司自己的产品开发者、微软信息技术工作者、微软的合作伙伴和客户以及微软咨询服务部门的经验，是当今指导应用软件开发的非常实用的规范。

为了清楚地介绍MSF，先讨论3个问题：应用软件开发概念的演进和相关术语、MSF评述及关于本书的结构。

MSF 0.1 应用软件开发概念的演进和相关术语

在当今软件技术迅速发展的情况下，许多对应用软件、软件工程及软件质量的传统看法都在发生变化，出现了许多新术语、新概念及新观点。MSF中有许多这类新观点。

0.1.1 软件项目的动态性

在早期的软件开发中，对一个较大的应用项目编出运行无误、功能齐全及性能较好的程序系统是很不容易的，既费工，又费时，周期也比较长(1～2年)。开发技术复杂，软件庞大，使得人们只想尽早完成，脱离没完没了的测试、修改、测试……枯燥乏味的工作；而且特别怕返工，总希望客户一次把需求说清，一次开发、测试、交付完成。交付后开发方就没有多大责任了，态度好一些的只要该软件在使用就服务到底。客户(使用)方一般都能体谅开发方做新的IT技术的难处，不太挑剔；而且在使用IT技术的初期，业务变化较慢，即使有变动要求也都迁就已使用的软件，不予变动，甚至由人为某些变动作预处理，以适应该软件。一直等到该软件实在落后，才提出需求，重新开发。现在IT业已成熟，有了大量的技术规范和支持工具，技术难点已并非最突出的问题。不再是客户迁就软件系统，而是软件系统为业务服务！客户方的业务随时日而变，才有市场竞争力。每当有新的IT技术出现，业务变更需求便有可能。为业务服务的应用软件，就必须是可连续变更的。客户方就可以在他们自己的领域内(银行系统、邮电系统、百货公司及出版系统)集中精力改进，做出更好的业务服务，占据更大的市场。不仅当前能做得较好，还能规划出可以预见的将来的更好的服务，例如，从零星的网络订货和送货上门，到全面的电子商务服务。

每当有了新的业务方案，应用软件系统便很快地被调整或开发出来，不再是过去的半年、一年或一年半，而是几个月、甚至几周。这种新观念导致了新的开发需求，软件支持的业务“变”是恒定的，“不变”倒是特例。一次交付是一个版本的终结，而不是该应用软件开发使命的

完结(当然合同可以一个一个地订)。这也就影响到开发方,一旦涉及了某领域,就应更加深入,协助客户做好他们的前景规划,而不是全然被动地"你提需求,我开发",以帮助客户做出应付万变的适变结构。

0.1.2 体系结构

软件变动的直接体现是程序代码改动,如果不根据设计直接改代码是十分危险的。设计是功能和性能的逻辑实现,只改物理实现(源代码),逻辑功能就会走样。早期的软件工程都一再声称"先设计,后编码"。当时所说的设计指模块/对象设计;现在的设计,是指整个应用软件的设计,包括增、删及换掉模块/对象,或改变它们的关系。同样,没有一个全局的观念,直接改设计也是十分危险的。因为模块/对象在实现程序逻辑时总会形成体系结构(architecture——不同层次上各职能模块/对象各司其职。它们的交互形成的结构就是体系结构),如果修改某部分而不计其余,这个有机的体系结构就有可能崩溃。人们终于认识到体系结构设计远比模块/对象设计重要。在软件变更不太频繁的时代,体系结构设计虽不可少,但远不如当今突出,而且过程式强耦合模块使我们在重设计体系结构上回旋余地不大。今日松耦合的面向对象模块使我们能方便地把不变对象和易变对象区分开来,聚合、集成比较简单,从而能获得较优的体系结构;因而,以体系结构为中心的软件开发逐渐成为应用开发的主流。

体系结构并不是软件编程以外的东西,一个应用系统就是一个体系结构(模块及其结构关系总是有的)。只是过去在实现模块分工、集成功能时自然形成;现在把注意的焦点转向以体系结构实现计算要求的功能。先以规格说明和接口(可以进一步归纳为体系结构描述语言ADL)表述所设计的体系结构,下一步才是构件(模块/对象)的设计与编码。软件开发人员确实要换一个思路:不要急于设计、编码实现构件,应先定义构件是什么,分析构件间的关系更为重要。这就是以体系结构为中心的软件开发。它特别适用于分布式对象系统,以体系结构实现业务逻辑,然后到各个异质节点上设计实现这个体系结构。业务有了更改,就改变体系结构的可变部分,重新在各节点上实现;反之,某节点有了更改(软、硬件升级),只是重实现该节点,对体系结构影响很小,甚至不会影响到业务逻辑。

为了定义、表述体系结构,人们做了大量研究。结果表明,体系结构是人们应付大型复杂软件极有力的工具。大的体系结构是由小的(轻量级)体系结构组成的。若采用标准的定型的小体系结构型(pattern),则构成大体系结构既快又不易出错,应用系统开发的复杂性一下子减少了一个数量级。只要背会了几十个型,实现一个庞大的应用系统就不会成问题。

体系结构有助于更加宏观、系统地看待软件。一个应用系统从纵向看不外乎是 3 个层次的体系结构的综合。硬件以上有技术基础设施(操作系统、网络协议、程序运行设施及支持语言的编程规范的设施等);第 2 层是实现业务逻辑的应用程序系统;第 3 层是各种用户界面。它们都是由各种不同的构件有机交互(形成体系结构)实现的。从横向看也是体系结构,例如,用户界面层可以采用两层的客户/服务器结构的客户端;也可以是带中间件(middleware)的 3 层结构的客户端;再如,业务逻辑层可以是层层调用的树状紧耦合的模块结构,也可以是松耦合的对象——消息结构,只要把各层次、各构件的接口定义清楚,大而复杂的系统不难分治。

以体系结构看待系统并不是新概念,历来如此。硬件就非常讲究体系结构,然而,对于早

期的、依附于硬件的、又只有一种层层调用的软件体系结构而言，是否讲究体系结构都不起什么大作用。今天的情况则相反，由业务导出软件体系结构，硬件实现软件所需的运作。解决方案最为关键，软件和硬件都是实现解决方案的手段，何况硬件越来越便宜，标准的软件构件也不难找到。

正因为传统软件开发不强调体系结构和技术基础设施，所以开发只是以应用程序的逻辑为中心，开发活动没有软件部署（将一个物理体系结构的应用系统落实到各站点的平台上），一切都隐含在选用的平台、操作系统和开发工具之中。既然都在一个选定的平台上，就无所谓部署了，需专门做的技术基础设施非常之少，甚至像界面自动生成工具（典型的技术基础设施）也都归于环境工具之中，不属于应用开发关注的论题。

0.1.3 基于模型的软件开发

在以体系结构为中心的应用软件开发中，为了控制应用系统的易变性、处理的复杂性，人们发现模型可以更加抽象、更加本质地描述体系结构。抓住了模型，才可以看到哪些变动是不影响程序逻辑的一次新实现，哪些变动影响了应用逻辑。业务演进就是体系结构的演进，更本质地是模型的演进。建模在应用程序开发的最初年代就有。早期的解题模型不过是几个数学公式（数学模型），按照它写 FORTRAN 程序非常方便。以后按照数据结构写算法，建模呼声似乎不太高了。其实选用数据结构就是定义数据模型，流程图就是算法模型。后来软件工程时代的数据流图（DFD），是基于数据流程的功能模型，E－R 图是数据模型，都是在利用模型，只是它们的表述有所不同（表示法不统一），有助于把软件做出来就行了。建模是软件做不出来不得已而为之的手段，不是目的。

现在情况则相反，模型是最主要的。不仅要建立模型，还须维护模型，随着软件的演进，须维护模型的演进。从业务过程模型映射为解决方案的体系结构模型，就是以体系结构模型实现了业务模型。体系结构是体系结构模型的一次例化实现。构件设计模型是体系结构的实现，构件是构件设计模型的实现。类/模块的模型又是构件的实现，最后按类/模块模型写出类/模块代码。这种做法和面向对象的技术一脉相承。面向对象的程序描述，定义的是类及其关系；而程序运行交互的都是实例对象，如果某个对象要修改，则重设参数，重新生成一个实例对象代替它。只有重大功能/性能修改时，才改类对象（及其关系）。类是实例对象的模型。同样，一个实例体系结构总要落实到分布式的某些站点上。如果某个站点更换或增、删，依据体系结构模型就可以审查是否可增、删、改，及怎样改对业务更有利。这就是对体系结构模型的维护。

当今软件开发有许多模型，不同的抽象层次上都可以有自己的模型，如概念模型、逻辑模型及物理模型；不同的开发阶段也可以有自己的模型，如分析模型、设计模型及小组建组模型；甚至动态的过程也有自己的模型，如自动机模型、状态模型及同步通信的汇合模型。

要维护一个模型，模型就必须是可表示的，不能寓于程序或计划中。模型可用图、表、数学公式及特定的文字表示。每种表示都有它的基本元素，如图元、表项、数学符号及它们组合的规则等。这些元素和规则的表示法和定义则由更高抽象的元模型（meta model）给出。元模型是描述模型的模型。

当今软件开发中元抽象的概念盛行，以上是因由之一。在数据领域，同一数据在分布式异质的不同站点上的表示是不同的。怎样让联网的机器协同工作呢？就靠元数据和元模型。每个具体站点的数据都是同一元数据的不同实例。这样就有了共同的理解：元数据是描述数据的数据。这些都是 Internet 普及后的常用术语。

0.1.4 软件过程

沿着时间轴去看每一件事物，都是过程，从该事物发生、发展、壮大、萎缩直到消亡。如果许多人一起开发一个软件，投入营运后，又有一批人维护、使用、直至废弃，称做软件过程。前者为工程过程，后者为营运过程。对于参与过程的人而言，过程是不同的人、不同的活动的序列；对于软件（人们劳动的对象）而言，过程的履行导致软件状态的改变，从无到有到完善到衰败到废弃。早期的软件工程以生命周期的各种活动刻画过程，并以瀑布模型的形式把顺序的生命周期活动作为开发过程模型，以致于给人以错觉。生命周期过程就是软件开发和营运的过程，也就是只有一种顺序的过程。

随着大量基于瀑布模型的软件工程项目的失败（＞70％），20 世纪 80 年代中期人们意识到，软件开发和任何复杂系统一样，是多次反复实践、认识、再实践、再认识的过程，于是提出了各种各样的迭代式过程模型。生命周期模型未变，开发过程模型变了。前者从状态的角度看软件过程；后者从工程角度看软件过程，而且认识到工程过程对工程产品的质量保证是至关重要的。一个工程产品没有若干关键的工程活动是出不来的。比如，没有需求分析无从谈设计，没有编码无从谈测试；但是什么时候做这种关键的活动，则是十分有讲究的。一点不差才是最优过程；差多了，其它关键的活动则无法进行。最极端的例子是大学教育过程：大学 4 年要学 42 门功课，按年限先学基础课，后学专业基础课，最后学专业课。当然，有些课程对先后不敏感，哪个年级学都无所谓。如果一个糊涂的教务处排课人把这 42 门功课混排，一年级上专业英语课，四年级上公共英语课……这个教育过程肯定是失败的，因为它违反了循序渐进的规则。再者，如果课程安排是循序渐进的，但课程的内容浅薄、陈旧或过深、过难，这种教育过程的结果也是欠佳的。

所以，一个工程过程的优劣取决于对以下 3 个问题的考虑。

- 要做成这个项目该有的活动是否都有了，还有哪些活动可使产品更好？
- 这些活动本身的质量如何？怎样才算做到家？——与所选技术质量标准有关。
- 活动安排是否恰当？顺序是否无冲突，完成后是否窝工。——与过程模型选取和计划安排有关。

工程过程不仅对工程产物质量影响巨大，对工程本身的成败也是至关重要的。优质工程意味着，按时、按质、按量（工程产品应有的功能和性能）且不超过预算完成；反之，则超预算，延期交工，低质量和功能不全。软件工程过程也一样，进度、质量、功能及成本 4 个指标需同时综合地达到，单方面追求一项最优必然导致其它项趋劣；为此，业界标准化组织制定了各种标准。软件开发时，若某项指标已达到标准，则可把精力投向其它方面。至于其它无标准的因素，则由双方协商，在合同中写明，如交付期、功能/性能特征的个数等。

0.1.5 相关标准的评述

软件工程是过程，按工程过程制定标准已成为共识。就是说一个企业，且不说做什么具体项目，只要做项目，该有的计划、管理、检测、评估、控制、包装、交付、培训及服务是否都有了。把这些活动和项目制作的关键活动(对软件项目而言是分析、设计、构造及测试)混在一起，形成过程。如果过程缺乏了某些重要的活动，则认为这个工程过程是不能保证质量的。

1. ISO9000

ISO9000 系列标准本是任何工业企业质量管理体系的标准族，就是从过程角度制定的质量标准。其中 ISO9000—3 和 ISO9001 特指软件行业。ISO9001 规定了 20 种必须实施的活动(大活动也叫做子过程)，包括 4.1 管理职责；4.2 质量体系；4.3 合同评审；4.4 设计控制；4.5 文件与资料控制；4.6 采购；4.7 顾客提供产品的控制；4.8 产品标志和可追溯性；4.9 过程控制；4.10 检验和试验；4.11 检验、测量和试验设备的控制；4.12 检验和试验状态；4.13 不合格品的控制；4.14 纠正和预防措施；4.15 搬运、存储、包装、防护和交付；4.16 质量记录的控制；4.17 内部质量审核；4.18 培训；4.19 服务及 4.20 统计技术。至于这些活动进一步的质量指标，依赖于各行业或企业自己的标准。例如，计划精确到[小]时、天、周、瑕疵等级和交付条件、文档的内容及种数……。

2. ISO12207

对于 IT 行业，ISO12207(1994 年)软件生存期过程给出了软件生存期全过程中的所有活动。传统的生存期活动(分析、设计、构造、测试及维护)只是它的主过程的一部分。ISO12207 还有大量有关管理、质量保证、文档及基础设施配置等子过程(共 12 个子过程，主过程只占 5 个子过程)，参见图 0-1。从图 0-1 中看出，开发软件编出代码的技术活动固然重要，但是距离完好的、可交付的工程产品，还差得很远。交不出合格的产品，在市场中没有效益，编程只能算做游戏而已。

ISO12207 的支持过程和辅助过程中的各项活动除“过程改进”之外，都要穿插在开发活动(主过程)之中。例如，编码完成之后要“确认”；每一阶段完成之后要“评审”技术问题和“审计”财务和进度；发现缺陷和问题则进行“问题解决”活动；“管理过程”、“文档”、“配置管理”及“质量保证”则贯彻开发始终。不是只实施一次，而是在项目主过程中不断实施。有的文献把主过程的技术活动称做关键活动域，把支持和辅助过程的活动称做伞形活动域，因其示意图像一把伞，如图 0-2 所示。

显然，图 0-2 是十分简化的示意图。有些活动或子过程是必须做完一个再做下一个的，如高层设计没有完成，无法做详细设计；模块编码没有完成，不能测试；构造完成，才能验证测试。有些活动插入的时间有弹性，例如，如果邀请用户参与开发全过程，开发过程一开始，“培训过程”就开始了，到产品交付后，再做集中培训，就简单一些；若用户参加不了，则从交付后开始集中培训。

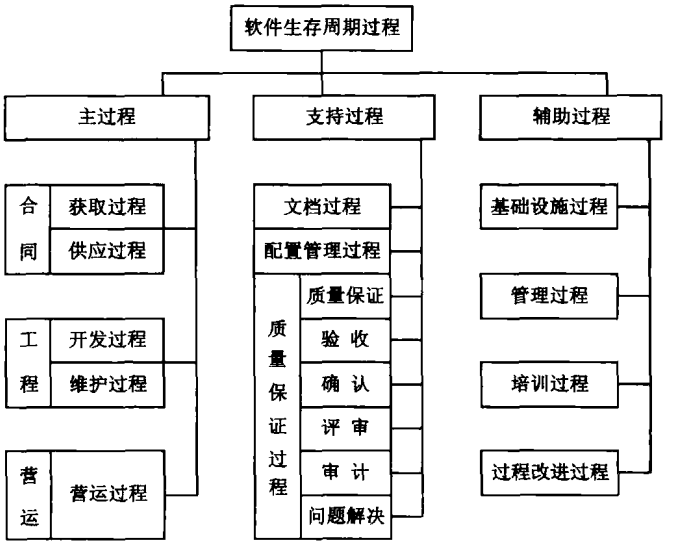


图 0-1 ISO12207 的过程分类

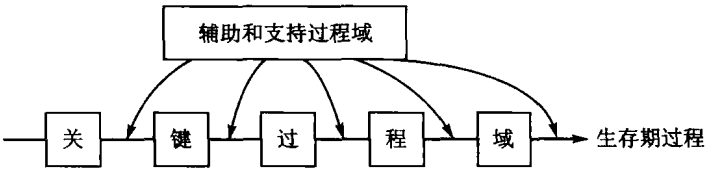


图 0-2 辅助和支持过程的活动插入关键域示意图

3. 过程成熟度模型

软件过程如此重要，以致于要以它作为评判一个软件企业有没有资质(格)做正规应用软件的标准。ISO9000 系列首当其冲，我国已在执行。通不过 ISO9000 是拿不到大项目的。谁也不敢拿几十、几百万元往一个没有保障的小作坊投资。作为软件企业，现在正在接受 CMM 标准。

CMM 的全名是能力成熟度模型(Capability Maturity Model)，是美国国防部委托卡内基-梅隆大学的软件工程研究所(SEI)研究的项目，旨在评定一个组织开发软件的能力。能力是指企业驾驭过程的能力，所以，俗称过程成熟度。CMM 1989 年被提出之后，立即风靡全球。因为“跑江湖”的承包人太多了，他们信誓旦旦地承诺，却没有组织良好过程的能力，不能按时、按质、按量交付软件。即使让他们索赔了，但延误的时间也是一笔巨大的浪费。虽然，美国计算机协会和国际 ISO 组织目前尚未承认 CMM 为标准，但它对行业的影响已经超过一般标准。目前 CMM 已经成为西方发达国家评价第三世界国家提供外包服务的软件公司的标准。印度是 CMM 执行得最好的国家。

CMM 把企业控制软件过程的能力分为 5 级：初始级(initial level)、可重复级(repeatable level)、可定义级(defined level)、可管理级(managed level)和可优化级(optimizing level)。这 5 种级别都是针对企业的软件过程的，并具体规定，有哪些管理内容才算达到某一级。