MIS 第3章

信息系统建设概论

在这一章里，我们将简要介绍信息系统的生命周期，包括信息系统的开发过程、实际运行和维护管理。目的是使读者从一个信息部门主管或项目负责人的角度体会一个信息系统的来龙去脉，了解各个阶段的任务，以及各个阶段之间的联系，而暂时不考虑复杂的技术细节。

3.1 信息系统建设是复杂的社会过程

3.1.1 信息系统建设的复杂性

进入21世纪，随着移动互联网和物联网的迅速发展，社会、购物、生活各类互联网应用百花齐放，“互联网＋”深入人心，信息系统应用变得无所不在、空前繁荣。技术革新催生了新的需求，也带来信息系统的更新换代。信息系统开发方法和工具一直随着信息技术的发展不断进步，但新问题层出不穷，建设道路仍然历尽坎坷。许多系统的效益远不如当初的承诺，半途而废、改弦更张时有发生，人们为信息系统建设的效率和成功率担忧。造成这种情况的原因，从根本上讲，是信息系统的多学科性、综合性等决定了它的复杂性是固有的、本质的，其发展必定有一个较长的过程，需等待各学科的成熟，技术人员和管理人员的知识需要拓宽，人员对信息系统建设过程的认识需要在实践中提高。

信息系统建设周期长、投资大、风险大，比一般技术工程有更大的难度和复杂性。这是因为：

1）技术手段复杂

信息系统是信息技术与现代管理理论相结合的产物，它试图用先进的技术手段解决社会经济问题。计算机硬件和软件、数据通信与网络技术、人工智能技术、各类决策方法都是当今发展最快的技术，是信息系统借以实现各种功能的手段。掌握这些技术手段，合理地应用以达到预期效果，是信息系统建设的主要任务之一。

2）内容复杂，目标多样

面向管理是信息系统最重要的特征。管理系统需要的信息量大覆盖面广，形式多样，来源复杂。一个综合性的信息系统要支持各级多部门的管理，规模庞大，结构复杂，非一般技术工程所能比拟。企业各部门员工和管理人员、不同用户群体或角色的需求不尽相同，甚至相互冲突，因而协调困难，不易求得各方面都满意的方案。有些需求是模糊的，不易表达清楚。对一般技术工程，往往可以通过具体模型或样品试验解决设计中的问题并完善设计，而信息系统的样品就意味着产品，在实际运行前几乎无法进行现场试验，系统开发中的问题只有投入运行后才能充分暴露。加之系统开发周期长，容易造成人力、物力和时间的

浪费。

3) 投资密度大,效益难以计算

信息系统建设,包括设备采购、系统开发和维护,都需要投入大量的资金。根据全球领先的信息技术研究和顾问公司 Gartner Group 的预测,2021 年中国 IT 支出预计将达到 3.04 万亿人民币,其中通信服务 1.27 万亿,设备 1.01 万亿,IT 服务 0.38 万亿,企业级软件 0.12 万亿,数据中心系统 0.26 万亿。信息系统采用大量的先进技术,但目前软件开发的自动化程度低,仍需要投入大量的人力进行系统分析、设计、编程测试和运维。信息系统建设是一种高智力的劳动密集型项目,简单劳动所占比例极小。这也是一般技术工程所不能比的。

同时,信息系统给企业带来的效益主要是无形的间接效益,不像一般技术工程取得的效益那样直接和容易计算。

4) 环境复杂多变

信息系统要成为企业竞争的有力武器,必须适应企业的竞争环境。这就要求信息系统的建设者必须十分重视、深刻理解企业面临的内外环境及其发展趋势,考虑到管理体制、管理思想、管理方法和手段,考虑到人的习惯、心理状态以及现行的制度、惯例和社会、政治等诸多因素。系统目标、功能既要适应企业当前的发展水平和能力,又要有足够的适应性,可以在一定范围内适应规章制度的变化,促进管理水平的提高,实现管理目标。这些更是不能同一般技术工程相提并论的。

3.1.2 信息系统开发是一个社会过程

将信息系统建设与一般技术工程相比较,可以看到,信息系统建设的困难不仅来自技术方面,还来自企业内外环境。影响信息系统成败的有体制、政策、法规、观念、技术等多种因素。技术不是唯一因素,甚至不是主要因素。

在相当长的一段时间里,人们把信息系统看作是计算机技术在某个组织的应用,认为信息系统开发是一个技术过程,视开发项目为“交钥匙工程”。用户认为开发是技术人员的事;开发人员认为用户应当陈述清楚他们的需求,由此出发开发系统,除此之外用户不要过多干预。用这种方式开发系统,往往造成双方误解,到“交钥匙”时,用户可能会提出“你开发的系统不是我所要的系统”,由此延误开发时间,浪费资源,或者因维护困难而使系统短命。

信息系统建设的实践,使人们越来越重视社会人文因素对信息系统建设的影响。信息系统是人机交互系统,其开发、维护都离不开人的参与。信息系统开发过程本质上是一个社会过程。从社会行动观点看,信息系统开发是人类活动的协调序列,是多种参与者的协作过程。在信息系统开发过程中,用户、系统管理者、系统分析师、系统架构师、软件工程师等利益相关者相互联系,相互影响。他们的通力合作是系统建设成功的基础。但是,由于这些人员知识背景、经历不同,会影响彼此沟通。通信的曲解是影响系统成功的隐患。更重要的是,信息系统建设不可避免地要改变某些业务流程乃至组织机构,这将影响某些机构和人员的工作方式、权力关系,引起机构之间、人员之间的利益冲突。有人会担心丢掉自己熟悉的工作,感到自己的传统地位和能力受到威胁;由于缺乏计算机知识,有人感到难以

适应现代信息系统的运行。这些担心常常造成系统开发的阻力。

信息系统不只是单纯的计算机系统，也是辅助企业管理的人机系统。人是信息管理的主体。因为人类活动是一种高级而复杂的影响因素，所以在有人参与并由人控制决策的社会系统中，往往会使本应理性的行为变得富有感情和丰富多彩。离开了人，再好的计算机系统，也不过是价格昂贵的装饰品而已。把信息系统的开发、应用和管理看作纯技术过程，许多问题永远得不到解决。只有从更深层次探讨，重视非技术因素，才有可能解决长期困扰人们的“软件危机”。

3.2 信息系统建设的一般方法

3.2.1 早期方法的不足

20 世纪 50 年代计算机开始用于管理领域。软件技术的进步，特别是数据库技术的出现，促进了管理信息系统的发展，20 世纪 60 年代出现了信息系统发展的第一个高潮。计算机的应用，使企业面目一新，提高了工作效率，增强了企业的竞争力。使用计算机成为一种时髦，企业和政府部门争相购买计算机，一些公司甚至在广告中写着“本公司用计算机进行管理”。有人称 20 世纪 60 年代是“计算机推销员的时代”。

然而这个高潮给人们带来的不总是成功的效益和喜悦，也带来了问题和教训。昂贵的计算机往往没带来设想的巨大经济效益，反而造成亏损，乃至企业倒闭。这种情况促使人们分析其中的原因。很多公司聘请系统分析专家对各种信息系统进行调查研究，总结经验教训，进而走上了较为顺利发展的道路。人们称 20 世纪 70 年代是“系统分析师的年代”。

出现这种情况的原因，从根本上讲是信息系统具有多学科性、综合性。信息系统的开发具有长期性、复杂性和风险性，需要科学的方法论作指导。造成系统开发失败有多方面的原因，如缺乏科学管理基础，领导重视停留在口头上，业务人员有顾虑甚至抵触。人们对信息系统的复杂性缺乏足够的认识，认为信息系统无非是“大程序”，缺乏开发信息系统的方法。

1. 目标含糊

信息系统是管理系统的一个子系统，它是为实现组织目标服务的。对于组织的目标没有明确的认识，对于信息系统要达到的目标没有明确的、恰当的规定，研发人员根据“想当然”来设计系统，危险性当然很大。对管理人员来讲，目标含糊表明对未来系统的状况没有明确的概念。这样，双方的想法必然产生差距。

2. 通信误解

研发信息系统，需要各级业务人员与技术人员密切配合。但这两方面的人员往往专业背景和经历极不相同，彼此不精通对方的业务，这就造成思想交流的困难，容易产生误解。而这种误解给系统带来巨大的隐患。俗话说“隔行如隔山”，在一方为常识性的、不言而喻的术语、规则，另一方却不一定理解。实际工作中常常遇到这样的情况，许多业务人员精通自己的业务，但不善于把业务过程明确地表达出来，他们觉得某项业务理所当然地就应该这么做，或者是凭经验和直觉就该这么做，而不是根据信息流或判断逻辑来进行。加上用户往往缺乏计算机知识，不了解计算机能做什么，不能做什么，因而更不可能用计算机技术

人员熟悉的术语介绍业务过程。而计算机技术人员因为缺乏用户方面的业务知识,不知道该问什么问题。这样,系统设计人员对用户的要求理解不透,有许多遗漏和误解。根据这种理解建立起来的系统当然有许多缺陷,甚至根本不是用户所要求的。

3. 步骤混乱

信息系统的开发是一项长期的复杂工程,各个工作环节之间有着内在的逻辑关系,超越某个阶段就会出现问题,造成返工和浪费。不经过深入的系统分析,只是根据对系统的肤浅理解就进行程序设计,这不仅不能保证各部分的正确衔接,而且肯定造成返工和重复劳动。本想早日完成系统开发,结果变成多次反复,旷日持久,欲速则不达。遗憾的是,把建立信息系统看成买计算机和编程序的现象,至今屡见不鲜。

4. 缺乏管理控制

信息系统的开发是一项复杂的系统工作,往往需要多方面的人员在较长时间里合作。研发期间常有人员、环境的变动。因此,缺乏计划性和缺少必要的管理控制势必使系统的建设涣散,难以协调,不能达到最终目的。

3.2.2 系统方法的应用

系统科学方法为人们提供了新的思维模式,是研究复杂系统的有效工具。钱学森曾指出:“系统工程是组织管理系统的规划、研究、制造、试验和使用的科学方法,是一种对所有系统都具有普遍意义的方法。”在信息系统领域,系统科学和系统工程方法的应用体现在以下几个方面。

1. 还原论与整体论相结合

还原论采用的是系统分解的方法,即将一个信息系统分解成若干个相对简单独立的子系统,每个子系统同理又可以分解成若干组成要素,这样自顶向下、逐步求精。只要研究清楚各个组成部分的性质就可以获得整个系统的性质,即整体等于部分之和。

而复杂性科学将复杂系统看作是一个整体,虽然为了分析的方便,会将复杂系统分解成若干部分来进行研究,但并非将整体看作是各个部分的简单叠加,整体的性质不是各组成部分性质的简单相加,各功能子系统之间的相互作用会产生新的性质,即整体大于部分之和。

如果没有整体论的指导,我们建设的信息系统可能是若干个“信息孤岛”的集合体。因此还原论和整体论指导我们在进行系统分解时,所分解的各组成部分是相对独立而不孤立的。

2. 微观分析与宏观综合相结合

微观分析的目的是了解系统的层次结构,而宏观综合的目的则是了解系统的功能结构及其形成过程。大规模的信息系统需要从微观描述过渡到宏观整体描述,如每个子系统和程序模块属于微观层次,系统整体架构则属于宏观综合。

3. 定性判断与定量计算相结合

信息系统是一个人机系统,计算机擅长进行定量计算,但是并不是对所有问题都可以

开发精确计算的程序，因此必须采取定量计算和定性分析相结合的方法。通过定性判断，可以建立信息系统总体及各子系统的概念模型，然后再尽可能将它们转化为数学模型或计算机模型。

4. 严格生命周期阶段与反复迭代相结合

在1.4.2节中介绍的霍尔三维结构的系统工程方法论，将系统工程活动分为前后紧密相连的七个阶段，每个阶段又分为七个步骤，根据这一方法论，我们将信息系统的生命周期划分为系统规划、系统分析、系统设计、系统实施、系统运行与维护五个阶段，每个阶段有明确的任务和成果，并强调上一阶段产生的输出是下一阶段工作的输入。

按照这种阶段划分的工作方式，系统开发得到严格管理和控制，大大提高了开发的成功率。但人类认识客观世界和解决管理领域问题的过程，是一个渐进的过程，是一个主动反复迭代的过程，因此信息系统的开发过程既要强调不同阶段任务的严格划分，同时也允许不断修正和迭代。

3.2.3 系统建模

1. 系统模型

系统模型是指以某种确定的形式(如文字、符号、图表、实物、数学公式等)，对系统某一方面本质属性的描述。系统模型是对原系统的描述、模仿或抽象。分析研究复杂系统问题，建模是一种基本手段。对于大多数研究目的而言，没有必要考虑系统的全部属性，因此根据实际需要可以从一个特定视角建立模型以描述系统某一方面的属性。

系统模型反映实际系统的主要特征，但它又高于实际系统而具有同类问题的共性。因此，一个适用的系统模型应该具有如下三个特征。

(1) 它是现实系统的抽象或模仿；

(2) 它是由反映系统本质或特征的主要因素构成的；

(3) 它集中体现了这些主要因素之间的关系。

系统模型根据抽象程度不同分为概念模型、逻辑模型和物理模型；根据时间的依赖关系可以分为静态模型和动态模型。对于同一个系统，研究目的不同，可以建立不同的系统模型，而要全面彻底地描述一个系统，通常需要使用多个模型。

2. 信息系统模型

信息系统模型大多数是图形和图表模型，本书的主要章节将展示如何创建和绘制信息系统的各种模型。每种图形或图表模型都有公认的标准符号、惯例和语法规则，这些符号和规则形成了一种特殊的表示语言，也称为建模语言。

信息系统的建模语言一直处于发展和完善之中。一些模型符号简单，目的单纯，没有严格的规范，不具备复杂的语言特性，因此表达能力有限，只能描述系统某一个方面的特性，比如实体关系图。另一些建模语言有复杂的语义、丰富的表达符号和多个模型图，能从各个视角一致性地描述系统，如统一建模语言(unified modeling language，UML)。

表3.1列出了信息系统开发中常用的一些模型，本书后续章节将陆续用到。

表 3.1　常用信息系统模型

模型名称	用　　途
业务流程图	描述不同职能部门业务活动分工和活动过程
数据流图	描述数据的产生、处理、存储和去向的信息处理模型
程序流程图	描述程序完成顺序、分支、循环等处理过程
实体关系图	描述系统中有价值的实体及其关系的数据模型
组织结构图	描述组织的部门及其从属关系的层次模型
模块结构图	描述软件功能模块及其调用关系的层次模型
判定表、判定树	描述决策条件及其行动关系
UML(类图、用例图、顺序图等)	描述软件系统结构及行为的一组模型
甘特图	描述项目任务及其完成日期的项目计划模型

3. 信息系统建模的作用

建立模型的目的不是复制系统的原貌,而是帮助人们更好地了解和探究复杂事物的本质。可以将信息系统工程和任何一项复杂的建筑工程相比拟,建筑工程在前期要完成大量的需求调研和设计工作,有了设计图纸后工人才能开始根据图纸施工。同样,信息系统在具体实施之前也需要绘制“蓝图”,例如,需要分析和设计系统的硬件结构、软件结构、信息处理流程和商务规则等诸多复杂和纷乱的问题,并以模型将工作成果表达出来。可以这样说,信息系统分析与设计的过程实际上就是建模的过程。信息系统分析阶段建立的是系统逻辑模型,以表达信息系统需求为主;设计阶段完成系统物理模型,以设计系统软硬件结构和程序流程为主;信息系统战略规划则是建立系统概念模型,描述系统最基本的框架。

建立信息系统模型有以下六个作用。

(1) 对复杂问题进行简化描述,帮助有关人员简单、直观、准确地了解系统本质。

(2) 建模的过程使分析师和设计师能更全面地研究系统,深思熟虑,减少遗漏,形成更成熟的方案。

(3) 各阶段产生的模型为后续阶段的有关人员提供了工作依据。

(4) 为项目各类人员提供了统一的交流工具,利于沟通和团队合作。

(5) 为项目验收和将来的维护工作提供了文档依据。

(6) 利用工具将模型映射为特定平台的可执行代码,减少开发人员工作量。

3.2.4　建立管理模型

模型致力于解决实际问题,因此必须保证模型的建立和模型的结果都符合实际领域知识和行为。由图 2.5 可以看出,信息系统的开发本质就是建立管理模型并转化为信息处理模型的过程。在信息系统的生命周期中,建立管理模型是系统分析阶段的任务,在系统设计及实现阶段中,相应的模型转换为技术方案。

管理模型是通过模型描述组织的状况,包括组织的静态特征、动态特征、业务流程、商务规则等,如图 3.1 所示。

通过建立管理模型,使项目经理和信息系统开发人员对实施的项目有比较规范的依据。本节仅就建立管理模型的主要内容作一简要说明。

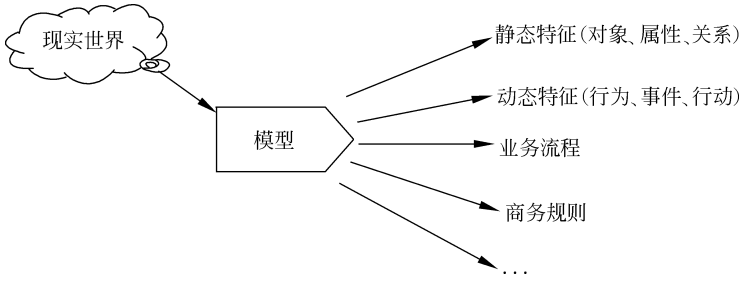


图 3.1　管理模型

1. 静态结构建模

建立管理模型，首先要对系统的静态特征建模。静态特征主要反映现实世界中“有什么”或“是什么”，如有关的对象、对象的属性、对象之间的关系等，组织机构的设置以及机构之间的关系等。

静态管理模型应用较多的是组织结构图和实体关系图(entity relationship diagram，ERD)模型。不同组织的业务不同，部门划分可以有直线职能式的结构，或者是基于产品或市场的分权管理的事业部结构，每个事业部下分各自的职能部门，以及便于项目管理的矩阵式结构。

【例 3.1】 图书馆组织结构图，如图 3.2 所示。

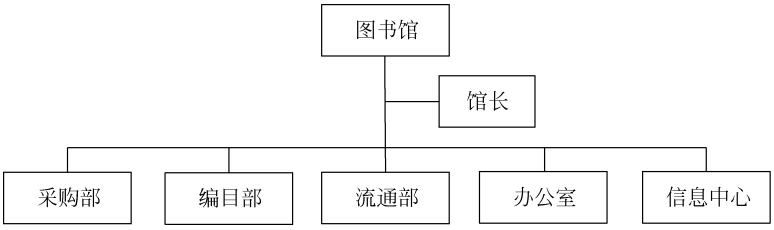


图 3.2　图书馆组织结构图

【例 3.2】 工厂库存管理的实体及关系。

图 3.3 是库存管理的 ER 模型，图中矩形框表示实体，内含实体名称和属性，直线表示实体间的关系，连线上的字母或数字表示关系的基数。

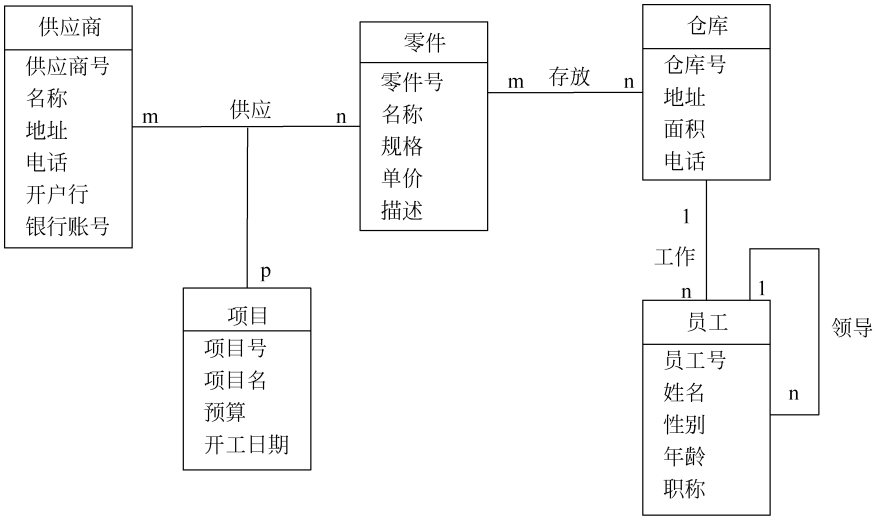


图 3.3　库存管理 ER 图

2. 行为建模

动态模型反映现实世界中实体的行为特点,包括实体行为、状态迁移、对象交互协作等方面的描述。

例如,人们经过图书馆的注册手续后便成为读者。读者可以反复执行借书、还书等一系列活动,通过注销而终止活动。对于每本图书而言,通过购入开始生命周期,由于丢失或下架而结束其生命周期。这些都可以通过绘制状态机图来描述。

【例 3.3】 会员积分的生命周期中的各种状态及其转换,如图 3.4 所示。

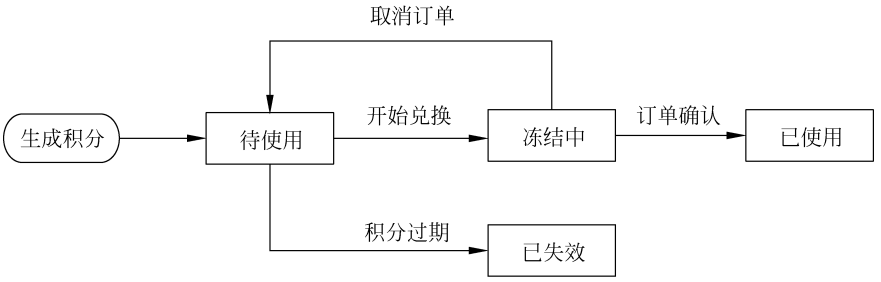


图 3.4 会员积分的生命周期

3. 过程建模

建立企业过程(business process)模型是信息系统分析的重要内容,也是企业过程重组的基础。本节简要介绍企业过程和企业过程建模的概念,具体的方法和工具在 6.1 节再详细介绍。

所谓企业过程是为实现某个预定产出的逻辑相关的一系列任务。企业过程有以下三个特征。

(1) 企业过程是跨越组织边界的。在按职能划分的企业里,部门对应的是与职能相关的任务,而企业过程包括多个任务,因而可能跨越部门,甚至跨越组织边界。

(2) 企业过程有层次性。企业过程有大范围的,例如企业全局的核心过程,也有比较细节的,例如具体的采购过程、产品开发过程。大范围的过程往往可以分解为若干子过程。应根据实际需要,考虑过程的层次。当子过程细化到具体工作岗位及活动时,通常使用“业务流程”这一说法。

(3) 每个企业过程都有输入和产出。这些输入和产出可以是人力、物资、能量、设备、信息。一个企业过程的产出可能是另一个过程的输入。

可以用语言,也可以用图形、符号对企业过程进行抽象的描述(见图 3.5)。这种描述不是企业实际过程无一遗漏的全部反映,而是通过抽象,提取一些重要的部件,以降低复杂度而又不失本质。由于叙述性的语言描述缺乏一致性的规范和结构,不利于不同人员之间相互沟通,人们采用规范化的图形符号,如流程图等来描述企业过程。我们把这种描述称为企业过程模型,而把这个过程称为企业过程建模。具体模型如业务流程图(在第 6 章有详细介绍)。

4. 商务规则

商务规则反映企业运作中的特定要求、必须遵循的约束和条件。这些条件和限制保证

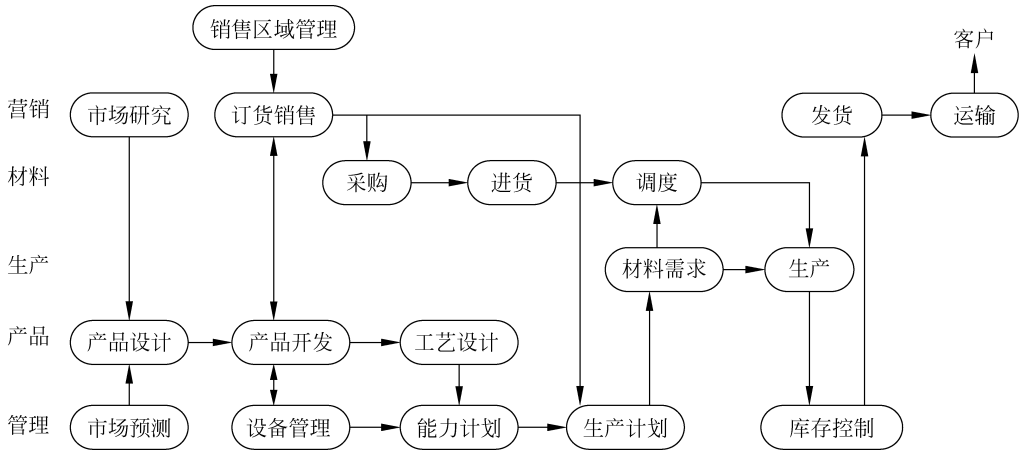


图 3.5　企业过程

了商务活动的正常运行，指明了商务活动中的各种要求，也建立了对商务活动的监督控制。

商务规则首先体现在数据的完整性约束上，包括属性的类型和值域、实体完整性、参照完整性。属性的类型和值域决定了该属性的基本数据特征。例如属性“月份”类型为短整型数字，取值范围为{1,2,3,…,12}。实体完整性意味着每个数据记录都应具有身份标识。例如“学生”这个数据会有一个非空的“学号”属性作为主键，学号的值可以唯一地指出某一个学生的数据记录。参照完整性反映数据属性之间的某种“存在性”关系，在数据库中体现为数据表之间关系的维护。例如一个派工任务可以由 1～3 名工人完成，一张处方单最多可以开 6 种药品。

此外商务规则体现在计算策略和决策行动上，如根据不同用户类型、订单总额计算优惠价格和积分，不同产品选择不同物流服务等。可以通过决策树、决策表这类模型描述。

在信息系统中，商务规则可以通过多种方式来集成。有些规则是比较明显的，而有些可能是隐含的，需要深入分析才能得出。商务规划集成的质量是决定信息系统品质的关键指标之一。

3.2.5　UML

UML 是由单一元模型支持的一组图示法。这些图示法有助于表达与设计软件系统，特别是采用面向对象方法构造的软件系统。本节简要介绍 UML 的起源、主要内容和发展趋势。

1. 什么是 UML

工程师们都需要绘制蓝图。一栋大楼的设计图在绘制时只需要几个人，施工时却需要上百的建筑工人；一架飞机设计蓝图时需要几十个航空工程师，但制造飞机可能需要几千人。绘制蓝图时不需要购买材料、不需要施工，只是利用图示模型事先规划好，比直接蛮干要便宜得多，而且可以利用蓝图来探究设计方案。

采用面向对象方法来分析和设计信息系统，最主要的图示模型应该反映问题域的对象/类及其结构，但仅从这一个视角来描述系统是不够的，强大的模型语言应能从各个角度

来建立模型,从而获取对系统的完整理解。例如既能反映系统静态结构,也能描述动态功能和过程;既可以表现抽象也可以深入到具体实现。另外,开发人员能够使用统一的图形符号对于行业内的交流共享也是非常重要的。20 世纪 90 年代中后期,面向对象领域的专家学者联合起来创造了一种通用的建模语言,并成为国际标准,这就是 UML。

2. UML 的主要内容

UML 作为一种可视化建模语言,由视图(view)、图(diagram)、模型元素(model element)和通用机制(general mechanism)等几个部分组成。其中视图表示系统的各个方面,由多个图构成。每个图使用了多个模型元素。在此基础上,通用机制为图做进一步补充说明,如注释、元素的语义说明。

UML 定义了以下几种视图,从不同角度反映系统。

(1) 用例视图(usecase view)。描述系统的功能需求,是最终用户、分析人员和测试人员看到的系统行为。该视图把系统的基本需求捕获为用例并提供构造其他视图的基础。

(2) 逻辑视图(logic view)。描述系统的基本逻辑结构,是问题的逻辑解决方案,展示对象和类是如何组成系统、实现系统行为的。

(3) 进程视图(process view)。用于描述系统性能、可伸缩性和吞吐量的设计,包含了形成系统并发与同步机制的线程和进程。

(4) 实现视图(implementation view)。用于描述系统组装和配置管理、表达软件成分的组织结构,包含用于装配与发布物理系统的构件和文件。

(5) 部署视图(deployment view)。描述组成物理系统的部件的分布、交付和安装,包含了形成系统硬件拓扑结构的节点。

系统模型中每一个视图的内容是由一些图来描述的。UML2.0 标准包含了 14 种模型图,信息系统建模主要用到其中的活动图、用例图、类图、对象图、顺序图、通信图、状态机图、构件图、部署图等。对整个系统而言,其功能由用例图描述,物理结构由构件图和部署图描述,软件的静态逻辑结构由类图描述,动态行为由状态机图、顺序图、协作图和活动图描述。

(1) 用例图(usecase diagram)。定义了系统的功能需求,它完全是从系统的外部观看系统功能,并不描述系统内部对功能的具体实现。在用例图中,参与者代表触发系统功能的用户或其他系统,用例代表具体的功能描述。

(2) 类图(class diagram)。描述系统的静态结构,表示系统中的类及其关系。

(3) 对象图(object diagram)。描述系统执行时一个特定时刻上的一组对象及其关系。对象图是类图的实例化。

(4) 顺序图(sequence diagram)和通信图(communication diagram)。均表示一组对象之间的动态交互关系,顺序图反映对象之间发送消息以及发送的时间顺序,通信图反映收发消息的对象的交互关系。顺序图和通信图是同构的,即两者之间可以相互转换。通信图在 UML1.x 曾命名为协作图(collaboration diagram)。

(5) 状态机图(statemachine diagram)。描述对象可能的状态和发生某些事件时状态的转换,强调对象行为的事件顺序。

(6) 活动图(activity diagram)。表述系统业务过程、工作流、用例或对象行为中各个活

动的流程，支持并行活动的表示。

（7）构件图（component diagram）。描述软件构件以及它们之间的关系，表示系统的静态实现视图。

（8）部署图（deployment diagram）。反映系统中软件和硬件的物理架构，表示系统运行时的处理节点以及节点中构件的配置。

在实际项目中，并不要求使用所有的图，例如计时图在嵌入式实时系统中普遍存在，但一些商业领域不常见。还有一些图之间是等价的，比如顺序图和通信图，通常不需要都画出来，或者可以利用建模工具软件直接在二者间进行转换。一般的建议是：在分析与设计过程中至少应该产生用例图、类图和顺序图。

图示语言只是一种建模工具。初学者不必花过多的心思琢磨 UML 如何做到精确细致。本节不对 UML 的各种图进行一一详述，具体使用章节会有详细介绍。读者可查阅本书附录 C 简介，也可以到 UML 官方网站下载最新版本。

3.3 信息系统的生命周期

任何事物都有产生、发展、成熟、消亡（更新）的过程，信息系统也不例外。信息系统在使用过程中随着其生存环境的变化，要不断维护、修改，当它不再适应的时候就要被淘汰，就要由新系统代替老系统，这种周期循环称为信息系统的生命周期。图 3.6 表示信息系统的生命周期以及相应的工作步骤。

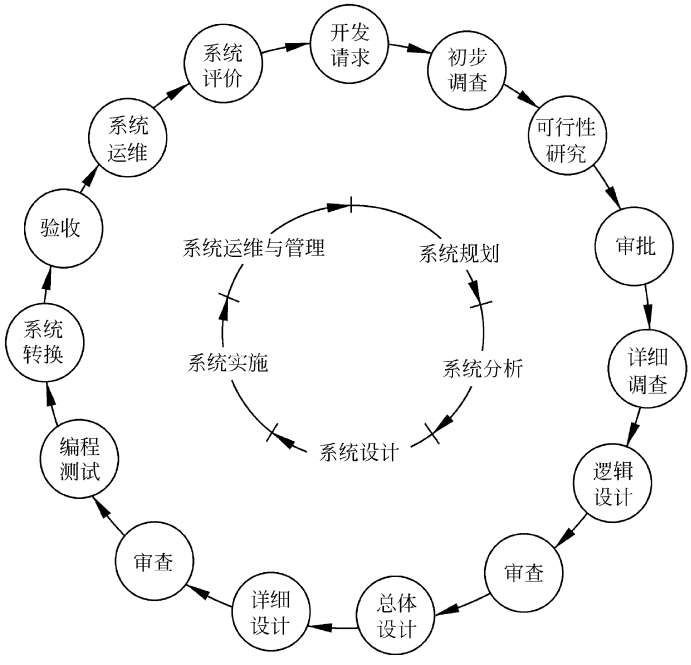


图 3.6 信息系统的生命周期

从图 3.6 可见，信息系统的生命周期可以分为系统规划、系统分析、系统设计、系统实施、系统运维与管理五个阶段。

3.3.1 系统规划阶段

系统规划阶段的任务是对企业的环境、目标及现行系统的状况进行初步调查,根据企业目标和发展战略,确定信息系统的发展战略,对建设新系统的需求做出分析和预测,同时考虑建设新系统所受的各种约束,研究建设新系统的必要性和可能性。根据需要与可能,给出拟建系统的备选方案。对这些方案进行可行性分析,写出可行性分析报告。可行性分析报告审议通过后,将新系统建设方案及实施计划编写成系统设计任务书。

3.3.2 系统分析阶段

系统分析阶段的任务是根据系统设计任务书所确定的范围,对现行系统进行详细调查,描述现行系统的业务流程,指出现行系统的局限性和不足之处,确定新系统的基本目标和逻辑功能要求,即提出新系统的逻辑模型。这个阶段又称为逻辑设计阶段。这个阶段是整个系统建设的关键阶段,也是信息系统建设与一般工程项目的重要区别所在。

系统分析阶段的工作成果体现在系统说明书中,这是系统建设的必备文件。它既是给用户看的,也是下一阶段的工作依据。因此,系统说明书既要通俗,又要准确。用户通过系统说明书可以了解未来系统的功能,判断是不是其所要求的系统。系统说明书一旦讨论通过,就是系统设计的依据,也是将来验收系统的依据。

3.3.3 系统设计阶段

简单地讲,系统分析阶段的任务是回答系统“做什么”的问题,而系统设计阶段要回答的问题是“怎么做”。该阶段的任务是根据系统说明书中规定的功能要求,考虑实际条件,具体设计实现逻辑模型的技术方案,即设计新系统的物理模型。这个阶段又称为物理设计阶段,阶段成果体现为“系统设计说明书”。

3.3.4 系统实施阶段

系统实施阶段是将设计的系统付诸实施的阶段。这一阶段的任务包括计算机等设备的购置、安装和调试,程序的编写和调试,人员培训,数据文件转换,系统测试,系统交付与转换等。这个阶段的特点是几个互相联系、互相制约的任务同时展开,必须精心安排、合理组织。

系统实施是按实施计划分阶段完成的,每个阶段应写出实施进度报告。系统测试之后写出系统测试分析报告。

3.3.5 系统运维与管理阶段

系统投入运行后,需要经常进行运行记录、维护、变更、审计评价等,记录系统运行的情况,根据一定的规格对系统进行必要的修改,评价系统的工作质量和经济效益。

各个阶段的主要成果及审核安排如图 3.7 所示。

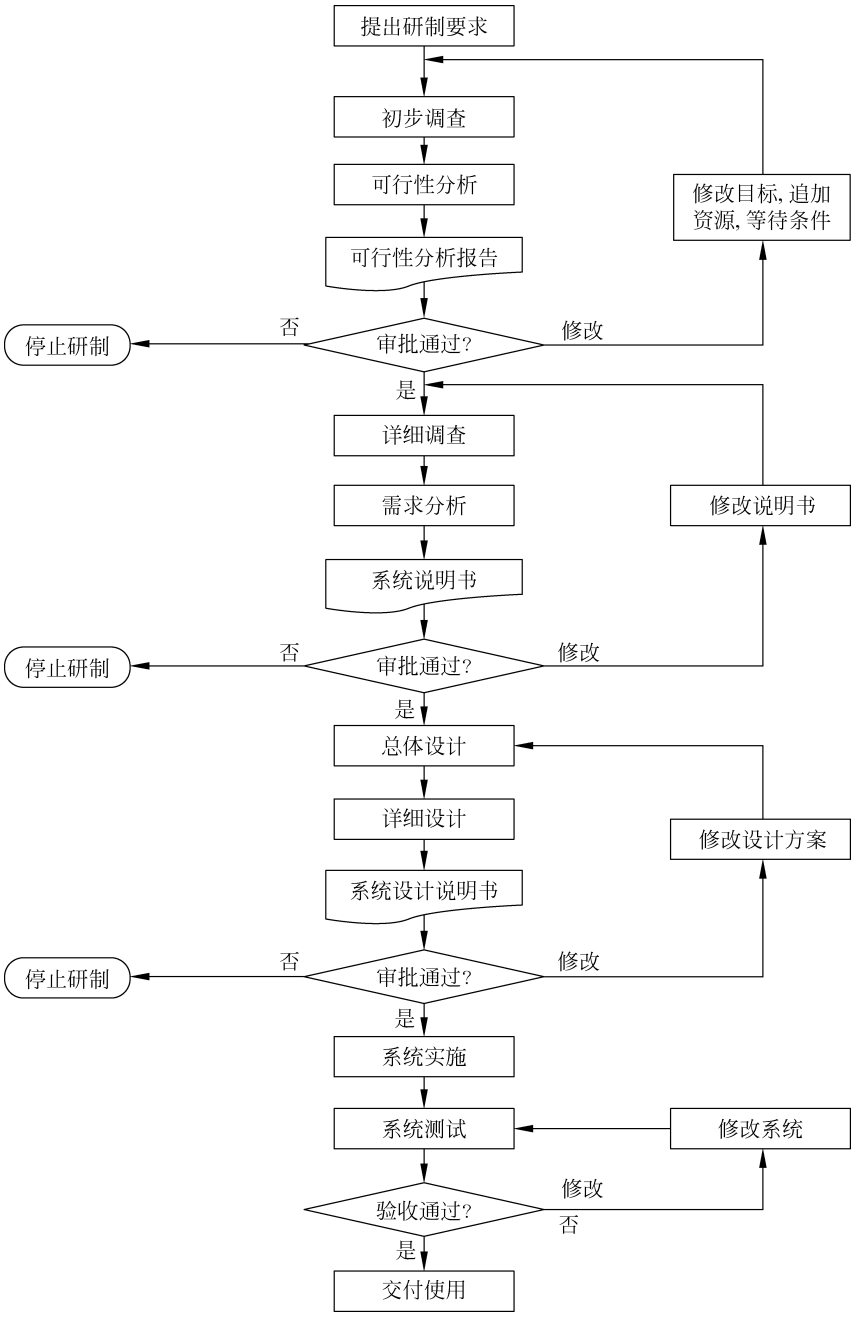


图 3.7 信息系统的开发过程

3.4 基于生命周期的开发方法

如 3.2.1 节所述，早期信息系统开发处于一种无序状态，经历软件危机后，开始采用系统工程方法论指导开发，即根据信息系统生命周期将整个开发划分为五个阶段，明确每个阶段的任务、任务的成果体现。最初有组织的系统开发就是遵照这些通用的阶段按顺序执

行各项任务。后来随着企业需求越来越复杂灵活,开发工具也越来越强大,不同信息系统项目的开发过程基于上述生命周期出现很多变种,从而衍生出各种开发方法,如瀑布法、快速原型法、迭代法、螺旋法等,这些方法按照不同的开发过程模型来完成各项开发活动。下面分别作简单介绍。

3.4.1 瀑布开发方法

在20世纪80年代前,一直采用的是严格按照生命周期阶段的开发过程,整个开发过程看起来就像是瀑布一样,稳定地向下依次经过这些阶段。每个阶段都有一个开始点和结束点,一旦到达下一阶段,通常不允许再回到上一阶段,正如瀑布不会向上倒流一样。瀑布方法的过程模型如图3.8所示。

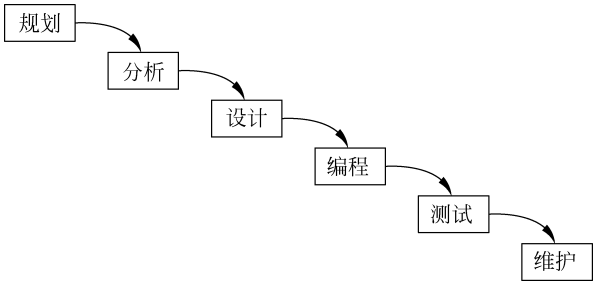


图3.8 瀑布方法过程模型

瀑布方法的最大特点就是阶段间严格的顺序性和依赖性,只有前一阶段完成,才能开始后一阶段;前一阶段的输出文档是后一阶段的输入文档。瀑布方法严格规定了每一阶段必须提交的文档和必要的审查验证。瀑布方法简单,易理解,易操作,它迫使开发人员遵守规范的方法和步骤,消除了系统开发的随意性,并且每一阶段对完成的文档进行严格审查,一定程度上保证了系统的质量。

但瀑布方法也有其不足和局限性。首先,瀑布开发方法需要在系统开发之初严格定义或明确说明用户需求,确定系统边界。从工程学角度看,这是十分自然的——解决问题之前必须明确要解决的问题是什么。然而对于信息系统建设而言,明确问题本身不是一件轻松的事,这正是系统分析的困难所在。其次,系统的推迟实现会带来风险,客户往往要等到开发周期的晚期才能看到系统的运行版本,这时若发现大的错误,可能引起客户的惊慌,其后果也可能是灾难性的。

因为信息系统开发是智力密集型项目,是人就不可能不犯错误,完全按照理想的自顶向下的瀑布方法几乎是不可能的,为此改进的瀑布方法增加了回退流程,即当后续阶段发现前面阶段的错误时,允许暂时回退到前一阶段,修正前面的工作成果后再回来继续完成中断的任务。

3.4.2 原型开发方法

正如瀑布开发方法中所提到的,并非所有的需求在系统开发之初都能准确地说明。实际上,用户往往善于描述其目标、对象以及想要前进的大致方向,而对于如何实现的细节却不甚清楚。建造一个系统,对所有参加者都是一个学习过程。用户通过观察和使用系统,

会推翻事先提出的需求，而提出新的需求，这种反复是不可避免且必要的，应该加以鼓励。

原型方法产生于20世纪80年代中期。其基本思想是：在投入大量的人力、物力之前，在限定的时间内，用最经济的方法构造一个系统原型，使用户尽早看到未来系统的概貌，在系统原型的实际运行中与用户一起发现问题，提出修改意见，不断完善原型，使它逐步满足用户的要求。原型方法的过程模型如图3.9所示。其中虚线部分就是该方法的核心活动——构造原型，该活动是一个和用户密切讨论不断反复的过程，最终确定的原型准确表达了用户需求。原型通常借助于快速开发工具或原型设计工具来完成，不适于直接投入使用，还需要根据实际开发平台环境进行再设计和编程实现。需要说明的是，如果原型只是完成系统部分核心需求的定义，那么在构造原型之后，全面的系统分析工作不可省略，图3.9未包含这种情况。

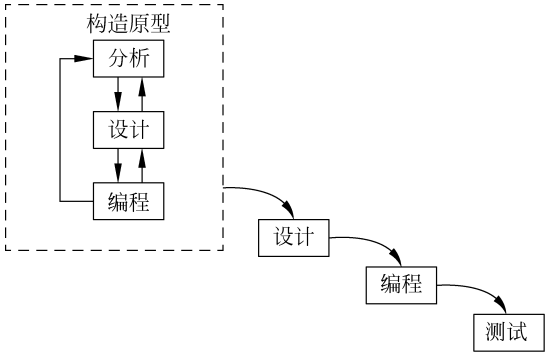


图3.9　原型方法过程模型

原型方法的优点很明显。

(1) 增进了用户与开发人员之间的沟通，启迪和发掘用户的真实需求。原型方法需要展示给用户可以实际运行的原型系统，使用户“看得见，摸得着”，可以很清楚地把他们的意见反馈给系统分析师。

(2) 用户在系统开发过程中起主导作用，随时提供现场的第一手资料，帮助开发者认识用户的真正需求。

(3) 降低开发风险，因为更有效地辨认用户需求，所以减少了开发人员对用户需求的误解，避免了较大偏离的情况发生。

(4) 可以帮助开发人员尽早验证系统架构、关键算法、人机交互等设计方案的有效性。

原型方法也有不足之处。原型法不如瀑布方法成熟和便于管理控制。由于用户的大量参与，也会产生一些新的问题，如原型的评估标准是否完全合理。原型的开发者在修改过程中容易偏离原型的目的，使用者在看到原型的功能逐步完备之后，以为原型可以联机使用了，反而疏忽了原型对实际环境的适应性及系统的安全性、可靠性等要求，便直接将原型系统转换成最终产品。这种过早交付产品的结构，虽然缩短了系统开发时间，但损害了系统质量，增加了维护代价。

3.4.3　迭代开发方法

上述原型方法只是一种需求验证的手段，如果将其思想应用到整个开发过程，使得每

个阶段的任务经过多次反复，或者将分析、设计、实施的周期反复多次，通过一次次迭代，不断在原来的基础上完善和修正，越来越靠近目标，这样的开发过程就称为迭代方法。

迭代方法产生于一种假设：一条直线一次性到达目的总是困难的。在信息系统开发中，系统的规模和复杂性日益增长，而市场竞争也对工期要求越来越紧，缓解风险和压力的方式是先提交一个有限的版本，细节部分逐步增加，即经过多次迭代后完成系统。

迭代方式有两种：增量迭代和进化迭代。

1. 增量迭代

增量迭代是将整个系统划分为多个小型的、功能相对独立的小项目(如子系统)，被称为一系列的迭代或增量。每一次迭代都包括了分析、设计、编程与测试一个完整周期，每个迭代周期完成一个增量，然后将它们集成，如图 3.10 所示。整个过程如同搭积木。

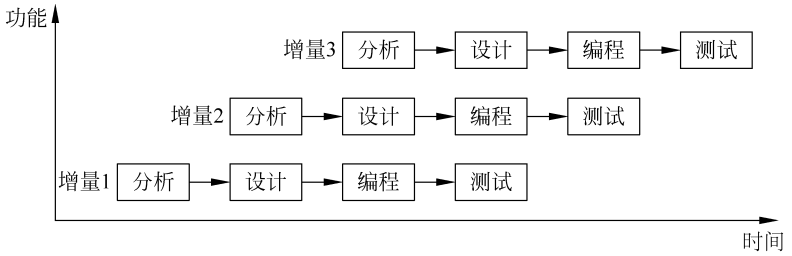


图 3.10　增量迭代的过程模型

2. 进化迭代

进化迭代与增量迭代的区别是每个迭代周期是对上一次迭代的演化和完善。如可以将一个软件功能的编程划分为多个迭代周期，每个迭代是对该功能的补充和进化。这个过程好比滚雪球。

迭代开发方法是目前应用最为广泛的开发过程，它以功能递增或进化的方式进行软件开发，不仅可以较快地产生可操作的系统，改善测试效果，而且分析师、设计师和程序员等不同技术人员可以实现并行化作业。此外，在每一轮迭代中，都可以把用户或开发人员的经验结合到不断求精的下一个迭代周期中，软件质量不断进步，降低开发总成本。

使用该方法较为困难的地方是迭代的定义，包括迭代长度(周期)。进化型迭代或小型项目可以一周迭代一次，增量型迭代或大型项目则可以 2～4 周甚至更长。项目组需要有经验丰富的架构师，否则很难规划出每次迭代的内容和要达到的目标，相关的交付件的验证和过程控制也需要投入较多精力。

3.4.4　螺旋开发方法

如果在每个迭代周期内加入风险分析，则会产生另一种过程模型：螺旋模型，如图 3.11 所示。这种模型的核心意图是将系统建设的生命周期分解为多个周期，多次开发完善系统原型，通过每个周期的风险分析，实现整个系统的风险控制。这里的原型不是用于验证的原型系统，而是最终要交付的成品系统。

图 3.11 中的四个象限分别表达了四个活动——制订计划、风险分析、工程实施、客户评估。沿着螺线自内向外每旋转一圈，便开发出一个更为完善的系统版本，最终得到所期望

的系统。若对于所开发的系统已经有了较好的理解，可直接采用瀑布模型（也可理解为单圈螺旋）。

由上面的介绍可以看出，不同过程模型各有利弊，通常要根据实际项目特点来进行选择。在前期需求明确的情况下尽量采用瀑布模型或改进型的瀑布模型。在用户无信息系统使用经验，分析人员技能不足的情况下一定要借助原型。在不确定性因素很多，难以提前估计和计划的情况下尽量采用增量迭代和螺旋模型。在技术难度较大、内容复杂的情况下采用进化迭代和螺旋模型。

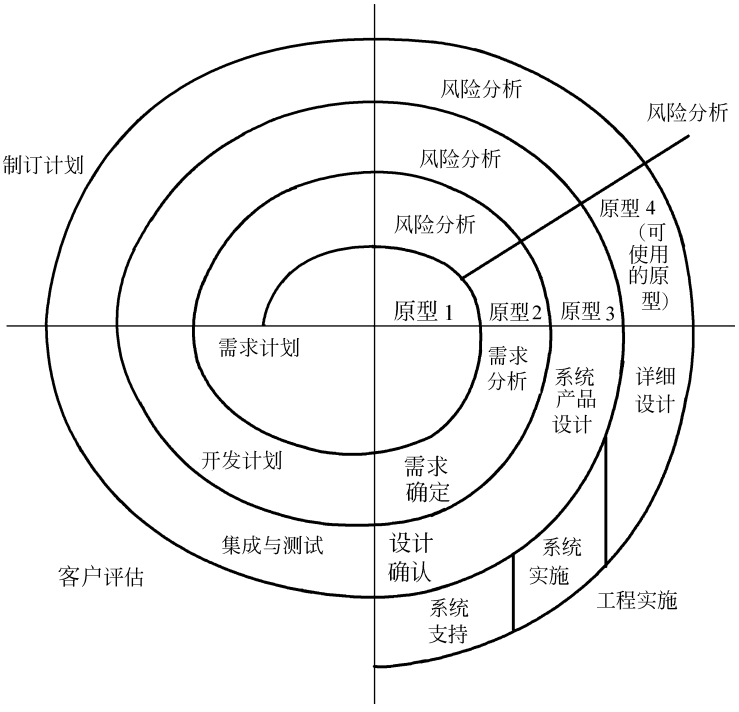


图 3.11　螺旋方法的过程模型

3.4.5　敏捷开发方法

1. 敏捷过程

过去几年中，人们提出了一种新的软件开发方法体系——敏捷方法学。由于传统的瀑布方法等过程存在繁文缛节的官僚过程，实施成本太高，并且对需求的变化反应不够迅速，因此敏捷过程越来越受欢迎。敏捷过程（agile process）是一系列轻量的过程模型的总称，它们致力于在无过程和过于烦琐的过程中达到一种平衡，以不多的步骤过程获取较满意的结果。

敏捷过程的主要思想可以从下面的《敏捷软件开发宣言》（manifesto for agile software development）中看出。

我们正在通过亲身实践以及帮助他人实践，揭示更好的软件开发方法。通过这项工作，我们认为：

（1）个体和交互胜过过程和工具。

(2) 可以工作的软件胜过面面俱到的文档。

(3) 客户合作胜过合同谈判。

(4) 响应变化胜过遵循变化。

虽然右项也有价值,但我们认为左项具有更大的价值。

敏捷过程也建立在迭代的基础上,倾向于使用较短的迭代周期,通常是一个月或更短。敏捷过程有很多代表模型,如极限编程(eXtreme Programming,XP)、Scrum、Crystal、特征驱动开发(Feature-Driven Development,FDD)等。下面对 Scrum 做简要介绍。

2. Scrum 框架

Scrum 这个单词来自于橄榄球运动,是指意外犯规或球出界后重新开始比赛、双方前锋低头争球的意思。创始人以此隐喻在软件开发中也应该使用橄榄球这种以团队为整体的工作流程,强调以人为中心,视勇气、承诺、尊重、专注和开放为其核心价值观。Scrum 并没有指定一个标准化过程,但提供了一套组织工作的框架,用于开发、交付和持续支持复杂产品。Scrum 框架由 Scrum 团队以及与之相关的角色、活动、工件和规则组成,见图 3.12。

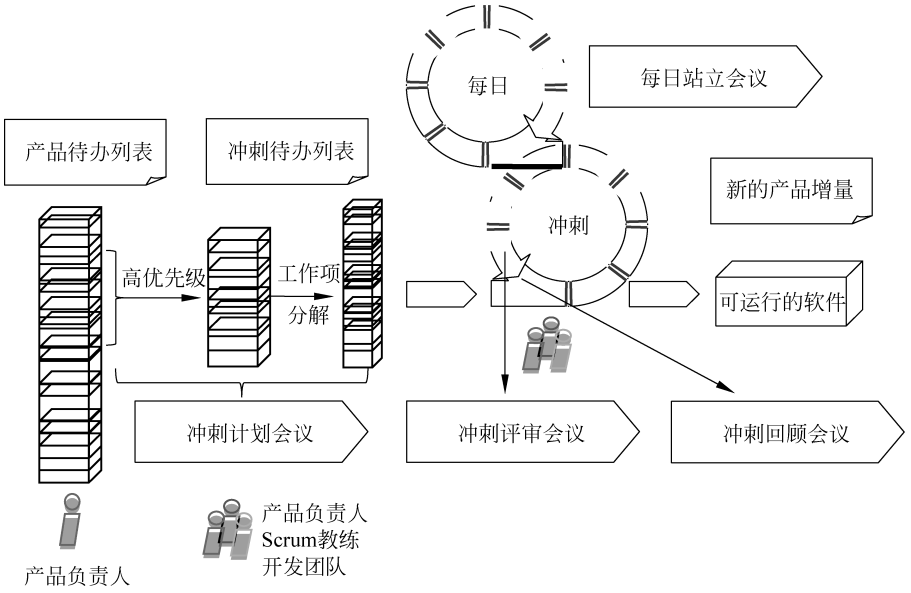


图 3.12 Scrum 框架

1) 角色

Scrum 团队由一名产品负责人、开发团队和一名 Scrum 教练(scrum master) 组成。产品负责人是负责管理产品需求,确定要开发什么、以什么优先级开发。开发团队包含各种专业人员,负责交付满足产品负责人要求的产品。教练负责指导团队在通用的框架上建立并遵循自己的过程,尽可能确保 Scrum 团队中的每个人都能理解目标、范围和产品域。

2) 活动

冲刺(sprint)是 Scrum 的核心,其长度为一个月或更短(迭代周期),一个冲刺过程内构建一个可用的和潜在可发布的产品增量。在整个开发过程期间,冲刺的长度保持一致。前一个冲刺结束后,下一个新的冲刺紧接着立即开始。

冲刺由冲刺计划会议、冲刺执行(开发)、每日站立会议、冲刺评审会议和冲刺回顾会议等活动构成。一个冲刺中要做的工作在冲刺计划会议中决定,计划由整个 Scrum 团队共同协作完成,体现团队成员间的信任和承诺。冲刺执行是开发团队为了完成冲刺计划而执行一些必要任务,冲刺执行完成后应产出一个产品增量,但不是全部产品。每日站会是开发团队的一个以 15 分钟为限的活动,为接下来的 24 小时的工作制订计划,同时通过检视上次站会以来的工作来调整团队协作、优化性能。冲刺评审会议在冲刺快结束时举行,对开发团队所交付的产品增量进行评审,并按需调整产品待办列表。冲刺回顾会议是 Scrum 团队检视自身,总结历史冲刺过程中的经验教训并适当调整,从而可持续改进开发过程和实践。冲刺回顾会议是在冲刺评审会议结束,下个冲刺计划会议之前举行。

3) 工件

产品待办列表(product backlog)是一份涵盖产品中已知所需每项内容的有序列表,它是产品需求变动的唯一来源。产品负责人负责管理产品待办列表的内容、可用性和排序。

冲刺待办列表(sprint backlog)是一组为当前冲刺选出的产品待办列表项,同时加上交付产品增量和实现冲刺目标的计划,它明确了下一个产品增量所需的功能以及交付这些功能所需工作的预测。

产品增量是一个冲刺完成的所有产品待办列表项的总和,以及之前所有冲刺所产生的增量的价值总和。

4) 规则

规则把角色、事件和工件组织在一起,管理它们之间的关系和交互。规则贯穿在整个 Scrum 框架的描述中。例如冲刺计划谁来制订、谁有权力取消冲刺、冲刺执行所需工作和预期不一致怎么解决等。

Scrum 采纳一种迭代、增量式的方法来改善预测和控制风险,已被成功用于开发软件、硬件、嵌入式软件、交互功能网络、自动驾驶,学校、政府等组织的管理运营,以及个体和群体日常生活或工作。另外一个敏捷过程代表——极限编程,与 Scrum 有很多相似之处,其提出的结对编程、测试驱动开发、持续集成、重构(refactoring)等很多做法深得人心,已成为行业标准或共识。

3.5 基于开发技术的开发方法

开发信息系统除了关注开发过程之外,更重要的是采用哪些开发技术来实现可运行的系统。本节从开发技术和建模的角度探讨信息系统不同开发方法的形成及其主要思想。

3.5.1 管理模型到信息处理模型

图 2.4 表明了信息系统是管理模型、信息处理模型、系统实现的基础条件这三者结合的产物。管理模型抽象描述了需要解决的管理问题(问题空间),而信息处理模型则回答信息系统将如何解决问题(解空间),这个求解过程中最关键的环节在于软件系统的实现。因为软件系统的状态比硬件系统的状态往往要复杂很多,只有找到控制和降低软件复杂性的方法,才能从根本上控制和降低信息系统复杂性。

软件结构是由计算机程序语言的特性决定的。几十年来,人们不断研究新的开发技

术,试图缩小计算机世界和现实世界之间的差距,从而让开发人员能以接近现实问题本质的方式去思考和描述问题,实现管理模型和信息处理模型的一致。

优秀的软件结构应具有以下特性。

(1) 能真实、充分地反映现实世界,包括事物和事物之间的联系,能满足用户对数据的处理要求。

(2) 易于理解,方便开发人员之间、开发人员与用户之间交换意见。

(3) 易于更改,当应用环境和应用要求改变时,能容易地对系统进行修改和扩充。

(4) 易于向计算机支持的数据结构转换。

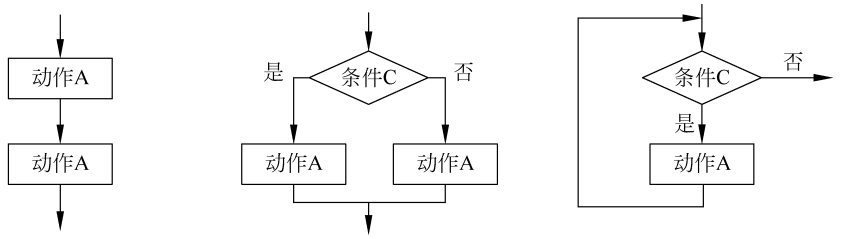
软件结构从简单到复杂,走过了从机器指令、语句、模块封装到类封装、再到构件和服务封装的历史发展过程,不同的开发技术和软件结构催生了不同的开发方法。

3.5.2 结构化开发方法

结构化方法论(structured methodology)是计算学科中一种典型的系统开发方法论。它采用系统科学的思想方法,从层次的角度,自顶向下地分析和设计系统,即抽象与分解。系统可用高级的抽象概念来理解和构造,这些高级的抽象概念又可用较低级的抽象概念来理解和构造,如此进行下去,直到最低层次的模块可以用某种程序设计语言的语句表示为止。

结构化方法产生于20世纪70年代中期。“结构化”一词出自程序设计,即我们熟知的结构化程序设计。在结构化程序设计出现之前,程序员按照各自的习惯和思路编写程序,没有统一的标准,也没有统一的方法。同样一件事情,不同的程序员编写的程序所占用的内存空间、运行时间可能差异很大。更严重的是,这些程序的可读性和可修改性很差,一个程序员写的程序,别人可能看不懂,修改更是困难,往往还不如重写。经过研究发现,造成这一现象的根本原因是程序的结构问题。

1964年,波姆(C. Böhm)和雅科比尼(G. Jacopini)提出结构化程序设计的理论,认为任何一个程序都可以用图3.13所示的三种基本逻辑结构来编制。戴克斯特拉(E. Dijkstra)等主张程序中避免使用GOTO语句,而仅用上述三种结构反复嵌套来构造程序。在这一思想指导下,一个程序的详细构造过程可按“自顶向下,逐步求精”的方法确定,即把一个程序分成若干个功能模块,这些模块之间尽可能彼此独立,用作业控制语句或过程调用语句把这些模块联系起来,形成一个完整的程序。这种方法大大提高了程序员的工作效率,改进了程序质量,增强了程序的可读性和可修改性,修改程序的某一部分时,对其他部分的影响也不太大。可以说这种方法使程序设计由一种“艺术”成为一种“技术”。



(a) 顺序结构　(b) 判断结构　(c) 循环结构

图3.13 程序的基本逻辑结构

人们从结构化程序设计中受到启发，把模块化思想引入到系统设计中来，将一个系统设计成层次化的程序模块结构。这些模块相对独立，功能单一。这就是结构化系统设计的基本思想。1974 年，康斯坦丁（L. Constantine）、斯蒂文斯（W. Stevens）和梅约斯（G. Myers）等在《IBM 系统》（*IBM System*）杂志上发表了《结构化设计》（*Structured Design*）论文，为结构化设计方法奠定了思想基础。

但是，结构化系统设计不能帮助系统设计人员建立一个直观的系统模型，使用户在实际得到并使用这个系统之前，就能够知道这个系统是不是他所需要的计算机信息系统。用户关心的是这个系统的逻辑功能，是否满足他的需要，是否能解决他要解决的问题。至于这个系统如何实现这些功能，并不是他最关心的问题。为了使所设计的系统满足用户的要求，在设计之前，先要正确理解和准确表达用户的要求，20 世纪 70 年代中期爱德华·尤顿（Edward Yourdon）等提出了结构化分析方法。而随着他于 1989 年所著的《现代结构化分析》（*Modern Structured Analysis*）的出版，该方法流行开来。结构化系统分析，强调系统分析师与用户一起按照系统的观点对企业活动由表及里地进行分析，调查分析清楚系统的逻辑功能，并用数据流图等工具把系统功能描述清楚。

结构化分析、结构化设计和结构化编程三种技术组成在一起成为结构化开发方法。由于分解是基于功能过程的，所以也称为“面向过程的方法”。

3.5.3 面向对象开发方法

面向对象（object-oriented）方法具有很强的类和对象的概念，因此能很自然地直观地模拟人类认识客观世界的方式，例如模拟人类在认知进程中的由一般到特殊的抽象功能，以及整体由部分元素组合而成等事物联系的分析功能。类的概念既反映出对象的本质属性，又提供了实现对象共享机制的理论根据。

面向对象方法是由面向对象程序设计技术（object-oriented programming，OOP）发展起来的，始于 1966 年的一种模拟编程语言 Simula。为仿真一个实际问题，引入了数据抽象和类的概念。几年后出现的 Smalltalk 语言被认为是第一个真正面向对象的编程语言。它吸取了 Simula 中类的概念，规定一切都是对象，程序设计以尽可能自动化的单元来进行，并开始用于实现基于对象的图形用户界面。随着 20 世纪 80 年代中期一些面向对象语言如 C++、VB 的出现，OOP 进入普及阶段。

OOP 的基本思想可以归纳为以下四点。

（1）客观世界的任何事物都是对象（object），它们包含一些静态属性和有关的操作。对象是一个整体，对外不必公开这些属性与操作。这就是对象的封装性（encapsulation）。

（2）对象之间有抽象与具体、群体与个体、整体与部分等几种关系，这些关系构成对象的网络结构。

（3）抽象的、较大的对象所具有的性质，自然地成为其子类的性质，而不必加以说明。这就是继承性（inheritance）。

（4）对象之间可以互送消息（message）。可以通过消息传送一个参数，消息也可以是使这个对象开始某个操作。

程序设计包括数据结构和算法（功能）两个方面，即信息的静态结构和对它的处理。对

象这个概念把这两个方面结合起来,使程序设计的思想方法更接近人们的思维方式。面向对象的程序设计为人们提供了更有力的认识框架。这一认识框架迅速地扩展到程序设计范围之外,相继出现了面向对象的数据库管理系统(object-oriented database management system,OODBMS)、面向对象分析(object-oriented analysis,OOA)、面向对象设计(object-oriented design,OOD)等技术,逐步合流形成一套完整的开发方法。

3.5.4 面向服务开发方法

服务主要指两个相关又有区别的概念:SOA 和微服务。

为了实现远程分布式的过程调用,在 20 世纪 90 年代出现了面向组件的编程技术,如 J2EE CORBA、DCOM 等。组件就是将程序进行封装,定义一些接口让外部调用,这些接口也称为应用编程接口(application programming interface,API)。客户端调用 API 时,客户端和服务器端之间以特定的传输协议进行通信,客户端不需要了解接口是如何具体实现的,也不需要引用服务器端的实现类。但由于组件技术标准不统一,导致不同技术实现的组件之间无法相互调用。2000 年开始,万维网联盟(world wide web consortium,W3C)基于互联网标准,陆续发布了简单对象传输协议(simple object access protocol,SOAP)和 Web 服务描述语言(web services description language,WSDL)协议,掀起 Web 服务热潮,从此依靠统一标准的 Web 服务,异构系统之间可以实现远程交互。不久,专家学者和各大厂商开始推广和普及面向服务的体系架构(service-oriented architecture,SOA),并共同努力制定了中立的 SOA 标准,SOA 开始进入了实施阶段。

面向服务的体系架构是一个组件模型,它将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的 API 联系起来。API 是采用中立的方式进行定义的,它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以一种统一和通用的方式进行交互。

SOA 是从业务角度出发考虑问题的,服务是可以独立封装的业务功能组件。SOA 方法提升了模型的抽象层次,它继承并加强了结构化和面向对象方法的通用软件结构设计思想,还增添了一些其他的主题,例如服务编排、服务库和服务总线中间件模式。

从概念上讲,SOA 中有以下三个主要的抽象级别元素。

1) 操作

代表单个逻辑工作单元的事务。执行操作通常完成数据的存取和加工。SOA 操作可以与面向对象中类的方法相似。它们都有特定的调用接口,并且能返回结果。

2) 服务

代表操作的逻辑分组。例如,如果将客户信用视为服务,则按照客户名称获得客户信用数据、建立信用记录、更新客户信用等就代表相关的操作。

3) 业务流程

为实现特定业务目标而执行的一组长期运行的动作或活动。业务流程通常包括多个业务调用。业务流程的例子有:批准一项贷款,本科生转专业,完成订单等。业务流程包括依据一组业务规则按照有序序列执行的一系列操作。操作的排序、选择和执行称为服务或流程编排。典型的情况是调用已编排好的服务来响应业务事件。

面向服务的分析与设计(service-oriented analysis and design,SOAD)不仅需要描述客户的业务流程,定义和编排服务,设计服务中每个操作的接口,还涉及企业架构,比如企业可以开发哪些服务组件,哪些服务可以公布供内部使用,哪些可以对外提供给相关部门或企业(如客户或供应商)使用。这涉及部门与部门之间、企业与企业之间的交互。总之,面向服务的方法最接近现实世界,管理模型和信息处理模型的一致性最高。

目前面向服务方法的最新研究和工程成果体现为微服务架构(microservice architecture)。微服务最早于2012年提出,是一种将一个单一应用程序开发为一组小型服务的方法。每个服务运行在自己的进程中,服务间通信采用轻量级通信机制。这些服务围绕业务能力构建并且可通过全自动部署机制独立部署。服务可用不同的语言开发,使用不同的数据存储技术,并且可以实现去中心化的数据管理模式。

SOA和微服务相同的地方在于都将服务作为一种进程外的组件,通过Web服务请求或远程过程调用(remote procedure call,RPC)机制实现应用程序之间的通信。但SOA的主要目的是为了企业各个系统更加容易地融合在一起,微服务主要为了解决大规模互联网应用系统的可扩展性和可靠性,同时控制团队规模,更易实行敏捷开发。

3.6 系统开发的组织管理

3.6.1 信息系统的企业发展模型

1. 诺兰模型

一个单位或一个地区的信息系统,要经历由初级到成熟的发展过程。诺兰(Nolan)总结了信息系统发展的规律,在1973年提出了信息系统发展的阶段理论,并在1980年进一步完善了这一理论,被称为诺兰模型。诺兰模型把信息系统的成长过程分为如图3.14所示的六个阶段。

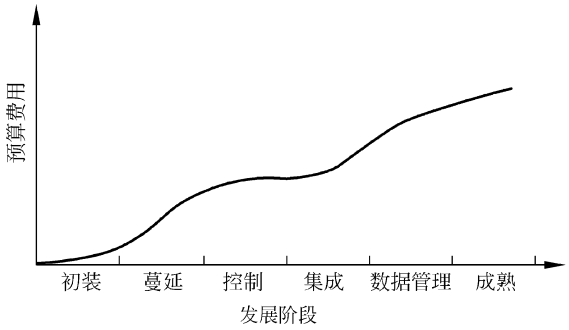


图3.14 诺兰模型

第一阶段:初装。

从单位购买第一台计算机用于管理部门就开始了初装阶段。在这一阶段,人们初步意识到计算机对管理的作用,有少数人具备了初步应用能力。

第二阶段:蔓延。

计算机初见成效吸引了人们,使信息系统扩散到多数部门,便进入了蔓延阶段。在这一阶段,数据处理能力发展很快,但很多问题有待解决,如数据具有不一致性、共享性差等。

这个阶段的投资迅速增长,但只有一部分系统取得实际效益。

第三阶段:控制。

解决第二阶段的问题,要求加强组织协调,对信息系统建设进行统筹规划。严格的控制代替了自由蔓延。这一阶段利用数据库技术解决数据共享问题。控制阶段投资增长较慢。

第四阶段:集成。

在控制的基础上,硬件重新连接,在软件方面建立集中式数据库和能充分利用各种信息的系统,这就是集成。诺兰认为前三个阶段属于"计算机时代",从第四阶段开始进入"信息时代"。这个阶段由于各种硬件、软件设备大量扩充,投资迅速增长。

第五阶段:数据管理。

集成之后进入数据管理阶段。因为当时美国还处在第四阶段,所以诺兰对数据管理阶段未作详细描述。

第六阶段:成熟。

成熟的信息系统应能满足组织中各个管理层次的要求,实现真正的信息资源管理。

诺兰模型总结了欧美发达国家信息系统建设的经验教训,具有普遍的指导意义。一般认为,模型中的各个阶段是不能跳跃的。实践证明,脱离实际的跨越,结果往往是跨而不越。在信息系统建设中,必须明确本单位所处的生长阶段,根据这个阶段的特点制定规划,确定开发策略,少走弯路。

2. 成熟度模型

从2000年以来,IT领域的研究机构陆续开发了各种成熟度模型,包括最早的软件工程能力成熟度模型(capability maturity model for software,CMM)、数据成熟度模型、信息安全成熟度模型等。成熟度模型可帮助企业领导者评估他们在技术领域、管理流程、业务、人员等各方面的成熟度,并建立路线图,提高成熟度水平,创造更高业务价值。

Gartner Group公司提出的IT基础设施与运营成熟度模型针对企业信息化建设整体成熟度给定了分级标准。与诺兰模型相似,成熟度模型定义了每个级别的特征,企业可以根据自己的企业状况以及各个等级当中的标志性技术,轻松地了解目前企业所处的阶段,以及发展到下一个阶段将要面临的问题,从而制定提高成熟度水平的路线图。根据经验,每个成熟度级别的过渡可能需要多年时间。读者可参阅相关资料了解各成熟度模型的等级评估标准。

3.6.2 建立信息系统的基础条件

经验证明,建立信息系统并使它正常运行并取得效益,必须具备一定的条件。这些条件有以下几点。

1. 领导重视,业务人员积极性高

国内外的经验证明,企业主要领导的重视和亲自参与,是成功建立管理信息系统的首要条件。一方面,管理信息系统是为管理服务的,只有最高领导最了解企业的目标和信息需求;另一方面,建立管理信息系统是一项复杂的系统工程,工期长,投资大,涉及面广,它的建立和应用可能涉及某些业务流程、规章制度,甚至组织机构的调整和改变,这些涉及全

局性的问题，只有最高领导亲自过问才能解决。

除领导重视外，业务人员的积极性也是一个重要因素。在系统开发阶段，需要他们积极配合，介绍业务流程，提供数据。系统建成之后，他们是主要的使用者和操作者。他们的业务水平、工作习惯和对新系统的态度，直接影响系统的使用效果和生命力。往往有这种情况：一个设计得很好的系统在一个企业失败了，但另一个类似的设计得不是很好的系统却在另一个单位成功了。关键的因素是人。

调动领导和业务人员的积极性，一方面要通过教育，普及信息系统的知识，提高他们的信息意识，消除误解；另一方面要吸收他们参加系统的开发，鼓励他们提出方案和建议。参与和交流是最有效的教育。

2. 有一定的科学管理基础

计算机的应用与管理水平的提高是相辅相成、互相促进的。管理水平的提高产生了对计算机的需求，计算机的应用又要求管理向更高水平发展。因此，建立信息系统，先要下决心研究管理问题，甚至下决心进行某些管理制度，乃至某些管理机构的改革。信息系统有各种形态，企业应根据实际管理水平，建立实用的信息系统，不要盲目追求整体性、综合性。一个战略目标不明确、管理制度不健全、数据不完整不准确的单位，首先要明确目标，健全制度，完善管理系统，使其科学化、完善化；否则，即使建立了管理信息系统，也不可能取得效益，假账真算，算得再快也毫无意义，反而为计算机的应用造成阻力。

3. 能组织一支具有不同层次的技术队伍

信息系统的开发和维护需要一支由各类专业人员组成的技术队伍，见表 3.2。仅有计算机技术人员是不够的，还应有经济管理方面的专家。

在信息系统开发过程中，系统分析是最困难的工作。系统分析师的知识水平和工作能力决定了系统的质量。缺乏称职的系统分析师是目前制约信息系统开发的重要因素之一。通常可由一些既懂业务又懂技术的人员组成系统分析小组，共同承担系统分析的重任。

4. 具备一定的资源

信息系统的建立和维护是一项投资大且有一定风险的系统工程。在工程正式开始之前，应有一个总体规划，进行可行性论证，对所需资源有一个正确的估计，制订投资计划，保证资金、设备按期到位。开发过程中要加强资源管理，防止浪费。

3.6.3 系统开发的准备工作

开发前的准备工作是建立领导机构。一个组织要开发管理信息系统，必须要该组织的主要负责人亲自领导，即“一把手原则”。我国的实践证明，主要领导人的重视与参与是管理信息系统成功的关键因素。只有主要领导人亲自组织，管理信息系统的开发才能顺利实现。

为了领导管理信息系统的开发工作，领导人应有运用现代管理科学提高企业管理水平的设想，具备信息系统的一些基本知识，了解信息系统的开发过程，善于组织队伍。推动信息系统开发的第一步是建立信息化委员会。信息化委员会是领导者的主要咨询机构，也是

系统开发的最高决策机构。其主要工作是确定系统目标,审核和批准系统说明书、系统设计说明书,验收信息系统。信息化委员会的成员应包括首席信息官(chief information officer,CIO)、首席数据官(chief data officer,CDO)、技术总监(chief technology officer,CTO)、有关部门负责人、有经验的管理专家、系统分析师等。委员会的主任由企业主要负责人担任。

信息化委员会可以下设项目管理办公室和各部门联络人,组织架构根据具体情况而定。人员可由各单位抽调,也可以外聘,或者内外结合。

系统开发中各类技术人员的职责和能力要求如表 3.2 所示。

表 3.2　系统开发中的各类技术人员的职责和能力

工作职务	职 责 和 能 力
系统分析师	同用户共同确定信息需求,编写系统说明书。应熟悉企业管理和信息系统开发过程,有较好的表达能力、与他人协同工作的能力。有根据信息流和组织目标改变组织职能的能力
系统架构师 软件设计师	设计信息系统,定义硬件、软件方案。能综合考虑系统的性能、安全性和质量,提供最佳系统架构和软件结构。应精通计算机硬件和软件。有较强的组织能力和决策能力
应用程序员	设计、调试计算机应用程序
系统测试员	计划和设计测试方案,并执行测试,记录和跟踪系统缺陷
程序维护员	维护现有程序
数据库管理员	管理和控制企业数据库
计算机操作员	操纵计算机设备
配置管理员	管理开发活动中的各项资产(源代码、文档、开发人员、软件版本等),组织和协调相关项目组成员实现变更控制
信息主管	规划信息系统的前景蓝图

除技术人员外,开发的各个阶段需要有业务人员的参加配合。开发的前期需要用户配合系统分析人员做好系统分析工作;后期需要用户承担切换、测试工作。为了使用户配合好开发工作,需要对用户进行培训,提出对他们的培训要求。图 3.15 是各开发阶段人力需求曲线。

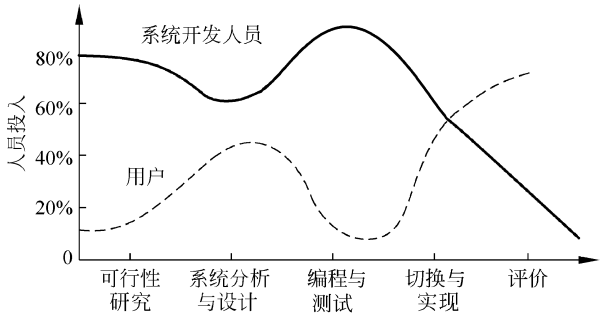


图 3.15　各个开发阶段的人力需求

3.6.4 选择开发方式

系统开发有多种方式，应根据资源情况、技术力量、外部环境等因素选择。不论采用哪种方式，都需要单位领导和业务人员参加。表 3.3 为四种系统开发方式的比较。

表 3.3 四种系统开发方式的比较

特　点	方　式			
	自行开发	委托开发	联合开发	购买现成软件包或软件服务
对分析、设计力量的要求	非常需要	不太需要	逐步培养	少量培养
编程力量的需求	非常需要	不需要	需要	少量需要
系统维护	容易	较困难	较容易	困难
开发费用	少	多	较多	较少

自行开发的好处是可以得到适合本单位的满意的系统，通过系统开发培养自己的力量。缺点是往往开发周期较长。自行开发需要强有力的领导，有足够的技术力量，需要进行一定的咨询。

委托开发从用户角度讲最省事，但必须配备精通业务的人员参加，经常检查、协调。这种方式开发费用较高，系统维护比较困难。

购买现成的软件包当然最省事。但要买到完全适合本单位的、满意的软件也不容易。有人说可以买到现成的计算机系统，但不能买到现成的信息系统。购买现成软件包需要有较强的鉴别能力。这种方式谈不上什么系统维护。购买软件最极端的方式是购买软件服务，一般以云计算的租用模式来使用软件，也称为 SaaS(software-as-a-service)模式。该模式允许用户在互联网上按期租用软件运营商的软件模块实现信息化管理，系统的软硬件部署、升级和维护都由运营商负责。它是云计算的一种类型。

联合开发对于培养自己的技术力量最为有利，系统维护也比较方便。条件是双方要精诚合作，自己有一定的系统分析和设计力量。

3.6.5 系统开发项目管理

信息系统开发具备一般项目的特点：它是一次性任务，有一定的任务范围和质量要求，有时间限制和进度要求，有经费资源的限制。因此，信息系统开发也是一类项目，可以用项目管理的思想和方法进行管理。

与一般技术项目相比，信息系统开发项目有以下特点：

(1) 目标不精确，边界较模糊，质量需求更多由项目团队定义。

(2) 信息系统开发项目进行过程中，用户的需求会不断被激发并进一步明确，导致项目进度、费用计划的更改。

(3) 智力密集，受人力资源影响大，项目组的结构、项目组成员的责任心和能力对项目的成功有决定性的影响。项目组成员不仅应具备一定的技术水平和工作经验，还要有良好的心理素质，善于与人共事，有高度的组织纪律性和团队精神，能放弃自己的某些“自由”去接受项目组的限制和约束，服从项目的严格管理。

信息系统项目管理的内容包括以下几点。

(1) 进度管理和控制。

简而言之,项目管理的过程就是制订计划,然后按计划工作。一个详细的计划包括:

① 确定执行项目需要的特定活动,明确每项活动的职责;

② 确定这些活动的完成顺序;

③ 计算每项活动所需要的时间和资源;

④ 制订项目预算。

计划制订之后,必须监控进度,确保工作按计划进行。若实际进展与计划有差异,必须采取纠正措施,并重新修订计划。计划评审技术(program evaluation review technique, PERT)是进度管理和控制的一项实用技术。在系统开发的准备阶段,可以制订较粗的计划,说明几个开发阶段的时间安排,绘制出网络图。随着开发工作的进展,再绘制分阶段的局部网络图。也可以用甘特(Gantt)图表示,如图 3.16 所示。

有关 PERT 的详细介绍,读者可参考运筹学、项目管理方面的书籍。

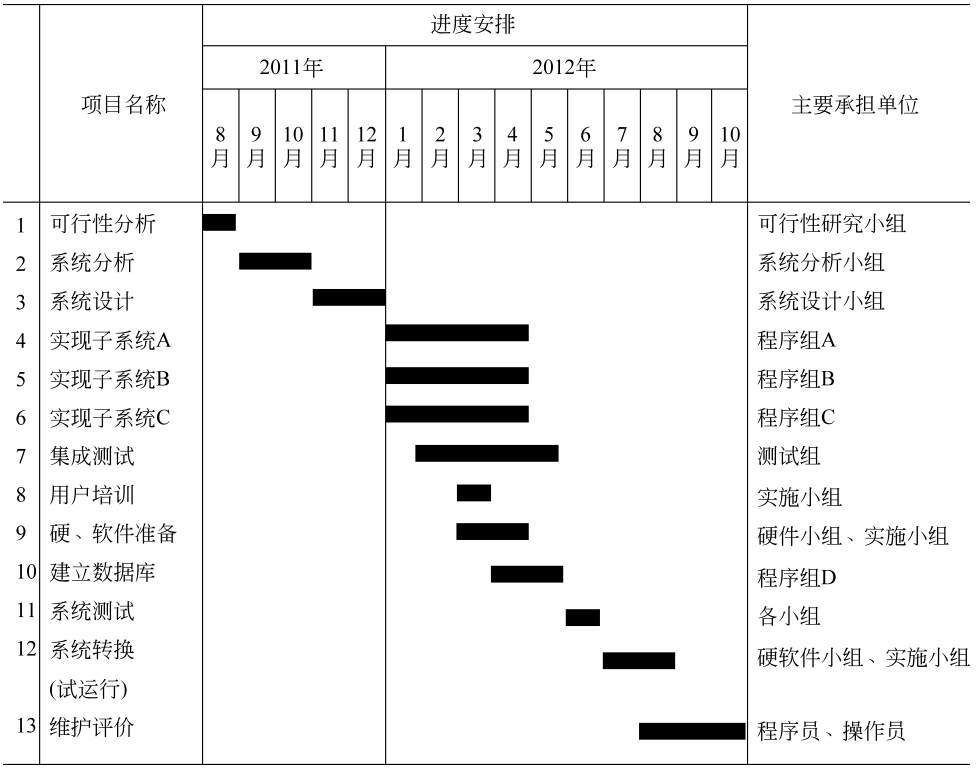


图 3.16 甘特图

(2) 经费管理。

经费管理是项目管理的一项重要内容,也是项目管理的有效手段。经费管理包括测算信息系统的成本、制订经费计划和成本计划的变更控制。

(3) 质量管理。

在现代质量管理中,质量是指“用户的满意程度”。信息系统要使用户满意,必须满足:

设计规格符合用户要求；程序要按照设计规格所说明的情况正确运行。质量管理包括质量计划、质量保证和质量控制。

(4) 文档管理。

文档是信息系统的生命线。信息系统开发的产品是软件，即程序加文档。对用户而言，程序是“看不见，摸不着”的。程序执行的对与错，用户只有看到结果才知道。没有规范的文档，程序不可能维护。

文档管理的内容包括文档管理制度化，文档编写规范化、标准化，维护文档的一致性，维持文档的可追踪性。

(5) 人员管理。

对于一个项目的成功，制订计划是必不可少的，但人员——项目经理和项目团队——却是关键所在。项目管理，归根到底是人员管理。项目负责人的素质，以及项目组的结构、责任心、能力对项目能否成功有决定性的影响。

3.7 信息系统开发工具

信息系统开发工具指在系统开发生命周期各个阶段帮助开发者提高工作质量和效率的一类软件，也称为 CASE(computer aided software engineering，计算机辅助软件工程)工具。在理解这一概念时，我们强调：

(1) 它是一种软件。

(2) 它是继高级程序语言之后，软件技术进一步发展的产物。

(3) 它的目的是在软件开发过程的不同方面给予人们不同程度的支持和帮助。

CASE 工具可以分为分析与设计工具、编程工具、测试工具、运维工具和项目管理工具。

1. 分析与设计工具

分析与设计工具统称为建模工具。

系统分析阶段需要使用建模工具来严格定义需求规格，并能将应用系统的逻辑模型清晰地表达出来。由于系统分析是系统开发过程中最困难的阶段，它的成功与否往往是决定系统成败的关键，因此分析工具应具备对建模的结果进行一致性和完整性检查，发现并排除错误的功能。设计工具是用来进行系统设计的，将设计结果描述出来形成设计说明书，并检查设计说明书中是否有错误，然后找出并排除这些错误。

分析与设计工具主要包括各种模型的绘制工具和快速原型的构造工具。常用需求分析建模工具有 Sybase 公司的 PowerDesigner、IBM 公司的 Rational Rose、微软的 Visio 等，它们可以用来创建业务流程图、数据流图、ER 图、UML 各种模型图、网络架构图、图形用户界面等。其中一些工具支持从模型生成部分源代码，以及从源代码生成模型的功能。

2. 编程工具

在程序设计阶段，需要为程序员提供程序开发环境，开发环境一般包括代码编辑器、编译器、调试器、图形用户界面设计器等。可以是单独的一种工具，也可以是将代码编辑、分析、编译、调试等合成为一体化的开发软件服务套件，这种套件就叫作集成开发环境(integrated development environment，IDE)，IDE 是目前程序员最常用的编程工具。

在以 Web 编程技术为主的工具中,依托不同的程序开发语言,有不同的开发平台和开发框架。基于 J2EE 应用的集成开发环境有 Eclipse/MyEclipse,IntelliJ IDEA 等,由不同厂家提供,很多是开源产品。.NET 则是由微软公司发布的强调多语言间交互的通用运行平台,.NET 应用开发使用 Visual Studio .net 集成开发环境。PyCharm 是采用 Python 语言开发 Web 应用的一款功能强大的 IDE。

除 IDE 工具外,还有很多开发框架也能帮助提高开发效率和速度。开发框架好比是一些软件"半成品",为应用程序的设计和开发提供了可重用的公用结构(组件和类),帮助程序员快速实现应用的定制,如 Spring、Django 框架。

3. 测试工具

软件测试历来是软件质量控制的主要手段,它是为了发现错误而执行程序的过程。测试工具应能支持整个测试过程,包括测试用例的选择、测试程序与测试数据的生成、测试的执行及测试结果的评价。测试工具不仅能够针对用户界面和功能进行自动化测试,还可以模拟客户机向服务器发送请求,测试并发用户数对性能的影响,数据量对性能的影响,网络配置对应用的影响,并对服务器各项资源进行监控等。

属于测试阶段的工具有静态分析器、动态覆盖率测试器、单元测试工具、自动化功能测试工具、自动化性能测试工具、测试报告生成器、缺陷跟踪和管理工具等。比如开源的压力测试工具 JMeter、自动化测试工具 Selenium 等。

4. 运维工具

系统运维的目的不仅是要保证系统的正常运行,使系统适应新的变化,更重要的是发现和解决各种事件和问题。

系统日常运行管理和实时监控程序、漏洞扫描工具、运维审计工具都属于运维工具。软件运行维护阶段的工具主要包括开发运维一体化工具、支持逆向工程(reverse engineering)或再造工程(reengineering)的反汇编程序及反编译程序等。

5. 项目管理工具

软件项目管理贯穿系统开发生命周期的全过程,包括对项目开发队伍或团体的组织和管理,以及在开发过程中各种标准和规范的实施。具体讲,有项目开发人员和成本估算、项目开发计划、项目资源分配与调度、软件质量保证、版本控制、风险分析及项目状态报告和跟踪等内容。

目前支持项目管理的常用工具有 PERT 图工具、Gantt 图工具、软件成本与人员估算建模及测算工具、软件质量分析与评价工具以及项目文档制作工具、报表生成工具等。比如 Microsoft Project 可以协助项目经理发展计划、为任务分配资源、跟踪进度、管理预算和分析工作量。

习题 3

3.1 小论文:结合现实案例阐述信息系统开发是一个社会过程。

3.2 对信息系统建模的目的是什么?你认为应该在哪些方面建模?

3.3 信息系统的开发可以分为哪几个阶段?各阶段的基本任务是什么?各阶段应提

供什么技术文档？

3.4 为什么说系统分析是开发信息系统最重要的阶段？这个阶段的工作困难在什么地方？系统分析师的职责是什么？

3.5 在系统开发中，用户起什么作用？为什么说信息系统的失败，主要是领导的失败？

3.6 基于生命周期的开发过程方法有哪几种？各自适应于什么类型的项目？

3.7 结构化方法的主要思想是什么？

3.8 面向对象程序设计和结构化程序设计有什么联系和区别？

3.9 什么是面向服务的体系结构？

3.10 购买软件包和购买软件服务有什么异同？请选择某个单位的信息系统，分析采用不同开发方式的利弊。

3.11 什么是 CASE 工具？选择一种 CASE 工具进行简明介绍。