"九五"国家重点电子出版物规划项目
计算机知识普及系列
2001编程宝典丛书 4

引人入胜

InstallShield6.X/VB/VC/Delphi

自由编程世界奇葩

安装程序设计与制作

北京希望电子出版社 总策划
张君 周洁等 编著

本光盘内容包括:

1. 本版书中示例安装程序代码
2. InstallShield 6.1 汉化程序
3. 安装程序使用的 API 函数介绍
4. InstallShield 内部库函数介绍

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

"九五"国家重点图书出版规划项目
计算机知识管理系列

2001 编程宝典 丛书 4

引人入胜

InstallShield6.X/VB/VC/Delphi

自由编程世界奇葩

安装程序设计与制作

本光盘内容包括：

1. 本版书中示例安装程序代码
2. InstallShield 6.1 汉化程序
3. 安装程序使用的 API 函数介绍
4. InstallShield 内部库函数介绍

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

安装程序是将应用程序完整、可靠地安装到终端用户计算机系统上的程序。任何一个功能完善的应用软件都应该提供相应的安装程序，这些安装程序可以使用各种方法进行制作。软件开发者尽管可以直接使用 API 制作安装程序，但是 InstallShield 目前是软件开发界广泛使用的安装程序制作工具，全世界绝大多数软件开发商都采用 InstallShield 以及相关工具进行产品安装、包装和发行，其安装技术已经成为程序安装事实上的工业标准。

本盘书对 InstallShield 最新技术以及功能和使用作了全面阐述，通过对其集成化环境（IDE）、编程语言、编程模板、对象以及内部库函数详细介绍，并配以丰富的应用实例，使读者很快制作出令人赏心悦目的专业级安装程序；同时还针对当今各种 Windows 环境下的安装程序制作技术以及 VC、VB、Delphi 开发出应用软件的安装程序制作过程作了比较详尽的介绍。本书还针对安装程序制作的各个环节，从工程规划到实施、从软件工程学到美学、从安装设计到人员培训都作了比较详尽的阐述，具有很强的操作性和实用性。

本盘书语言简明扼要，内容丰富，涉及面广，是计算机编程人员、开发人员的好伴侣，也是 InstallShield 使用大全。本书面向各个层次软件开发、设计、编程人员，可作为个人或软件公司应用程序产品发行方案的技术参考用书，也可以兼作安装程序设计参考手册和 InstallShield 使用手册。

本光盘含本版书中所有示例安装程序和作者开发的 InstallShield 6.1 汉化补丁程序，以及安装程序使用的 API 函数、InstallShield 内部库函数介绍。

系 列 盘 书：2001 编程宝典丛书（4）
盘 书 名：引人入胜 InstallShield 6.x/VB/VC/Delphi 安装程序设计与制作
总 策 划：北京希望电脑公司
文本著作者：张君 周洁 等编著
CD 制 作 者：希望多媒体开发中心
CD 测 试 者：希望多媒体测试部
责 任 编 辑：苏静 刘晓融 陈河南
出版、发行者：北京希望电子出版社
地 址：北京中关村大街 26 号，100080
网址: www.bhp.com.cn
E-mail: lwm@hope.com.cn
电话: 010-62562329,62541992,62637101,62637102,62633308,62633309
（发行和技术支持）
010-62613322-215（门市） 010-62531267（编辑部）

经 销：各地新华书店、软件连锁店
排 版：希望图书输出中心
CD 生 产 者：北京中新联光盘有限责任公司
文本印刷者：北京双青印刷厂
开本 / 规格：787 毫米×1092 毫米 1/16 开本 30.25 印张 708 千字
版次 / 印次：2001 年 1 月第 1 版 2001 年 3 月第 2 次印刷
印 数：4 001-6 000 册
本 版 号：ISBN 7-900056-39-4/TP・38
定 价：50.00 元（1CD，含配套书）

前　言

自从计算机诞生以来，人们对计算机软件无论从开发还是从使用上来说都充满了热情。特别是随着 PC 机的普及，应用软件的大众化、普及化、可视化以及人性化给软件业带来了飞速的发展。在这种环境下各种软件开发技术应运而生，层出不穷，为软件业的快速发展不断提供了动力。同样针对不同开发技术的专著、参考手册以及使用指南，如雨后春笋般的出现。但是纵览这些指导书，很难寻觅一本有关程序安装技术的指导性书籍，众所周知，无论采用什么工具开发出的应用软件最终都要完整地安装到终端用户的计算机上，怎样才能把这种安装过程做好，这样各种程序安装技术和工具应运而生了。本书的作者在长期从事各种软件的设计和开发过程中，深切感到国内有关这方面的资料很缺乏，同时针对目前对于安装程序制作技术的迫切需求，我们在相关开发经验的基础上，经过参考广大一线程序员的经验以及国外相关资料，编写了这本书。

本书对当今各种 Windows 环境下的安装程序制作技术作了比较详尽的介绍，同时针对目前最流行的工具 VC、VB、Delphi 开发出应用程序的安装程序的有关制作过程作了阐述。本书重点介绍了目前国际上最流行的，被誉为“万能”安装程序制作工具—InstallShield（本书主要以 InstallShield6.1 为主同时兼顾 InstallShield5.x），通过对其集成环境（IDE）、编程语言、编程模板以及编程对象详细介绍，并配以大量丰富的应用实例，使读者很快制作出令人赏心悦目的专业级安装程序。另外，本书还针对安装程序制作的各个环节，从工程规划到实施、从软件工程学到美学、从安装设计到人员培训都作了比较详尽的阐述。

全书共分四个部分十六章。第一部分从整体上介绍了当前安装程序制作技术，其中第一章对程序安装技术的整体概述，包括软件设计的工程学、美学的介绍；第二章针对程序安装的 Windows API 和工具作了比较详细的介绍和比较。

第二部分介绍了怎样使用 InstallShield，其中第三章介绍了 InstallShield 的开发环境以及如何使用该工具进行制作安装程序；第四章给出了一个快速上手的应用实例；第五章对 InstallShield 各种使用技巧进一步进行介绍。

第三部分为 InstallShield 的高级应用，即如何深入应用该工具提供的方法、手段，通过简单的脚本编程，制作出令人赏心悦目的安装程序。其中第六章介绍 InstallShield 脚本语言；第七章介绍了 InstallShield 脚本事件；第八章介绍如何使用 InstallShield 模板；第九章介绍了 InstallShield 对象。

第四部分为安装工程设计与实例，具体介绍了安装程序的规划、实施过程，以及在该过程中涉及到的各个环节。其中第十章介绍安装工程，第十一章针对安装过程中出现的各种文件处理和控制进行说明，第十二章介绍了安装过程中对目标系统的控制以及注册，第十三章针对安装程序界面进行说明，第十四章介绍了安装程序的测试和发行，第十五章介绍了 3 个实际的安装程序实例：VC、VB、Delophi 开发的应用程序，第十六章介绍了在安装程序制作过程中出现的各种常见问题及相应解决对策。

另外，附录 A 提供了 InstallShield 的保留字，附录 B 提供了 InstallShield 的错误码，附录 C 提供了相关的网络资源。

这四个部分是按照由浅入深的次序进行讲述的，因此从头到尾进行阅读本书是最佳学习路线。对于以前没有接触过安装程序和 InstallShield 的读者，必须按照本书的编写结构进行阅读，这样才能掌握安装程序设计技术；如果读者熟悉 InstallShield 操作，可以把重点放在第三、四部分以进一步熟悉 InstallShield 高级应用技术和安装程序具体实现过程；对于想了解 InstallShield 以及安装程序基本技术的读者，可以单独阅读第一部分，不过对于第一部分，建议所有读者应该仔细阅读，因为这一部分是各种安装程序制作工具的基础。

在阅读本书之前，建议读者先有基本应用软件开发基础。虽然安装程序本身并不要求软件技术的支持，但由于安装程序制作的对象毕竟是开发出的应用软件，因此，有一定编程经验的读者可以阅读得更好，如果读者有 C 语言基础更好。

本书突出以实例为主，便于读者通过阅读此书尽快掌握 InstallShield。书中实例从易到难，从简到繁，逐步深入，涉及了安装程序制作的方方面面（不仅针对 InstallShield），书中全部实例都包括在随书光盘中。

本书的全部编写工作由张君、周洁两人共同完成，同时周涛讲师、李明工程师也各为本书献上精彩的章节，全书由张君负责统稿。在本书的编写过程中，还得到了许多一线程序员和同行的帮助，作者表示深深的谢意。特别地，得到上海交通大学朱子述教授（博士生导师）、陈洪亮教授、金之俭副教授以及上海交通大学研究生院在物质和精神上的支持。另外，我们还要衷心感谢北京希望电子出版社的刘晓融老师，如果没有她的热情鼓励与支持，恐怕也就没有本书的出版。

由于软件技术的更新速度快，相应的安装制作技术不断发展，尽管我们做了最大的努力，本书仍难免存在一些不足之处，希望广大读者批评指正。

编者

2000 年 11 月

目　录

第一部分 程序安装技术

第二部分 InstallShield 高级应用

第三部分 安装工程设计与实例

第一部分 程序安装技术

第 1 章 安装程序概述

软件既是一项工程又是一种产品。如果把软件看作是一项工程，那么必须要按照工程的方法进行设计、实施，也就形成了所谓的软件工程。所以作为一项完整的工程项目，其中必须包含把项目的最终产品完整、有效地交付给用户这一过程。怎样才能无缝地完成这一过程，并得到用户的认可，以及尽可能减少其带来的双方之间的摩擦，已经成为软件工程设计和实施阶段的重要任务。

软件同时又是一种产品。既然是一种产品，其必然要符合产品的一系列特性。其中最重要的就是产品的包装。我们常常会有这样的经验：对于针对同一对象的几种产品，性能差别也不大，往往吸引我们注意力和购买欲望的并不是产品自身的某些特性（相对与其它产品），而是这种产品的包装。这一现象充分说明了人的认知心理的一种趋向。我们开发出来的软件产品在市场可能存在着许多同类型的产品，怎样才能使我们的软件产品可以从众多产品中脱颖而出，吸引更多的注意力和购买力，充满更强的竞争力，这是当今软件公司面对的一个重要问题。当然解决这一问题要涉及诸多方面的因素，但有一点很重要，即在提高软件质量基础上，给产品一个好的外壳。

我们都很清楚，在 GUI（Graph User Interface）环境下各种软件的界面设计已经成为软件开发的重点，但是 GUI 中 Interface 的意义不仅指界面，更重要指的是接口，一种与用户方的接口，其中包括了软件的界面，当然把软件的最终产品无缝地交给终端用户也是非常重要的内容。怎样才能做好这个接口呢，现在通常的做法是给软件一个“外壳”，这一外壳不仅要满足软件工程的要求，还要在外观上吸引人，充分迎合人的认知心理趋向，通过这一“外壳”把软件的最终产品安装到终端用户方的计算机上，这一过程即程序安装，俗称“打包”。

1.1 打包的概念

打包（Package）即通过采用一系列的方法和手段，把应用程序和相关的文件集中起来，形成一个可执行的程序包（一般为 Setup.exe）的过程。打包过程不是一种文件的简单堆积、集中的过程，也不是开发的文件经简单地拷贝集中的过程，也不是一种采用某种压缩、解压缩算法制作文件的过程。那什么才是真正的打包呢？

从打包的字面意思来看，上面提到的几种文件集中方式似乎都满足打包的要求，但是如果你采用其中的一种方式制作一个程序“包”，那不妨把这个“包”测试一下，到其它计算机上安装，看看运行效果如何。

试一试　做一个简单的应用程序（用 VC、VB 或 Delphi 以及其它开发工具），然后把编译出的可执行文件通过软盘或其他方式拷贝到其它计算机上，执行该应用程序，看运行效果？

估计试下来的效果，多半不是很理想。为什么会出现测试过程中的这样或那样的问题呢？具体原因说起来很复杂，但从根本上来说，上述测试中的“包”并不是真正意义上的包，因为其不具备可执行性这一基本特征，也不满足经打包过程制作出来的程序包的基本要求。

1.1.1　程序包的基本要求

“打包”制作出来的程序包要满足以下基本要求，只有满足以下要求的程序包，该打包过程才是真正意义上的打包过程。

程序“包”的基本要求：

- 可执行性
- 简单化
- 可靠性

如果一个打包过程制作出来的程序包满足以上三个基本要求，该过程就是一个合格的打包过程。那么什么叫作程序“包”的可执行性、简单性以及可靠性呢？

1. 可执行性

可执行性是基本要求中的核心要求，是程序“包”必须满足的要求的要求，也是其它两个基本要求实现的前提和保证。

可执行性的具体含义指的是制作出来的程序“包”可以在目标计算机上运行以及经过安装后，应用程序可以在终端用户的目标计算机上运行。这一要求看起来很简单，但实际做起来还是有一定难度的。因为程序“包”的可执行性具体包括以下方面的内容：

(1) 程序包的可执行性。这一点就不用细说了，如果程序包不能运行，那么打包的意义又何在呢？

(2) 应用程序的可执行性。打包的最终目的是让应用程序在目标计算机上运行，执行相关的操作，满足用户和系统的不同需求。所以无论采用多么先进多么有效的手段进行应用程序的打包，必须保证这一目的的实现。否则所做的任何努力，无论采用多么令人赏心悦目的安装界面以及多么丰富的包装外壳，都意味着徒劳。

(3) 保证运行环境的可执行性。为了保证应用程序的可执行性，在安装程序制作过程中，必须保证应用程序运行的终端用户计算机上有相应的运行环境。如果没有这种环境，安装程序必须有创建这种运行环境的能力；如果存在这种环境，安装程序必须有识别和注册这种环境的能力。这就对安装程序提出一种“智能化”的要求，这一点对于采用 RAD（Rapid Application Development）工具开发的应用程序尤为重要，如 VB、Delphi 之类的工具。

(4) 保证系统可恢复的可执行性。Windows 环境下的应用程序的用户都很清楚，当系统不再需要某一个应用软件时，我们通常将其卸载。卸载过程通过应用程序带的卸载程序或通过 Windows 的控制面版中的【添加/删除程序】进行操作来完成。经过这样的卸载过

程，必须保证系统在应用程序卸载后，可以可靠、安全地运行；确保系统中不存在任何由该应用程序装来的“垃圾”。

2. 简单性

简单性指的是操作的易用化，这也是 GUI 环境下的一个基本原则。对于安装程序，简单化也就使用户无需复杂的设置和初始化，只要简单地阅读一下安装说明书便可把应用程序安装到目标计算机上，同时应用程序可以运行完成该应用软件提供的大多数功能。即使安装程序在安装过程中需要某种选择和设置，也应该以某种方式提示给用户相关的信息，即做出某种选择或设置后，将会出现什么样的结果，会对应用程序和系统有什么样的影响。这样用户经过简单的确认，可以把应用软件的一部分或全部安装到目标计算机上。

当然安装程序的简单化并不能解决所有问题，这种简单化必须保证是在安装程序的可执行性的基础上而实现的。通常对于小型的应用程序，这种简单化基本可以做得很好；但对于大、中型软件工程而言，现场安装调试往往要花很长时间，诸如配置各种服务器、网络环境、硬件接口以及各种相关的设置（这时软件公司往往要派专人去进行现场安装）。因此在这种环境下安装程序的简单化必须为其可执行性服务。

3. 可靠性

同样安装程序的可靠性也必须在保证安装程序的可执行性的基础上，做到可靠性安装，这一要求同时也保证了安装程序的可执行性。安装程序的可靠性主要要做到应用程序和系统的双可靠性。在安装过程中，可能要对系统做某些修改，也有可能针对不同的系统应用程序的安装做相应的“智能化”调整，这就要求安装程序要有高度的可靠性。典型地，安装程序要适应不同的操作系统，在 Windows 95/98、Windows NT 3.51/4.0、以及 Windows 2000、Window Me 环境中，安装程序必须能识别各种环境，并采用不同的安装程序包进行安装；同时也必须检查环境是否满足要求，例如 VB 开发的应用程序在 Windows NT 中，必须要求 NT4.0 为 Pack3 以上，这时安装程序在安装过程中必须判别并及时把系统信息反馈给用户，以便于实施相应的措施。

可靠性是通过大量地安装测试来保证的。作为一个合格的应用程序，或者说一个合格的软件产品，在交付到市场或用户之前，必须经受大量的测试，其中包括代码测试、模块测试以及系统集成测试。安装程序测试是系统集成测试的一个主要部分，也是整个测试过程的最后一个环节。在具体测试中，往往要选用一台几乎是“裸机”的计算机进行测试，通过采用多种运行环境的测试后，基本上可以说该应用程序测试通过了。对于比较大的软件公司，在这一步完成之后，往往屏蔽软件中某些功能，形成所谓的β版软件投入市场，供用户测试使用，并通过用户的反馈，做出相应的修改，从而最终形成正式产品投入市场。关于安装程序测试的具体情况请参见第 15 章。

1.1.2　“打包”的起因

随着软件技术的发展，特别是在 Windows 环境下的软件开发技术有了突飞猛进的进展。虽然 Microsoft 提供了数以千计的 Win32 API（Application Programming Interface），但是任何有过直接使用 Win32 API 进行程序开发经验的程序员一定明白，这个过程简直

如同噩梦一般，大量 API 的直接调用以及错综复杂的界面和消息编码已经耗尽了程序员大量的宝贵时间，至于在追求完善、友好的用户界面那就更是烦中添烦，所以至今为止，很少见到一位完全采用 Win32 API 进行一个工程项目的程序员，如果真有这样的程序员，我们且不说这个项目做得如何，单从编程的角度来看，此程序员可称得上是一位大师级的人物了。

鉴于这种情况 Microsoft 以及 Inprise 公司先后推出了 Visual C++, Visual Basic 和 Delphi 等集成开发环境。这些可视集成化（IDE）环境提供了强大的界面设计工具，做到所见即所得的效果；同时还把 Win32 API 封装成类库、组件和控件，这样使程序员从繁重的编程中解放出来，把有限的时间投入到系统的设计和更完善的人机界面上，同时大大缩短了软件的开发周期，这无疑是软件开发史上具有里程碑意义的事件。但是采用这种可视化集成环境会带来一定严重的问题。在这种环境中开发各种各种软件，集成化环境已经自动配置好相应的系统运行环境，我们无需关心应用程序低层的具体配置。但是一旦把应用程序转移到没有这种环境的计算机上运行，那么很自然一定无法正常运行。对于这样的问题，不可能在目标计算机上再重新安装一套 VC，VB，Delphi，而且用户也不允许这样做。因此软件界就提出一个打包方法来解决这个问题，即把用以支持运行环境的各种文件（各种 dll，ocx，sys，drv 文件）与应用程序以某种方式集中后制作成一个 Setup.exe 程序，然后在客户方运行，把应用程序的运行环境在目标计算机上建立起来，这样应用程序在某种程度上就做到了无缝移植。这是安装程序制作的基本起因。

另外，应用程序往往利用安装程序进行系统的设置和应用程序的初始化，这又是安装程序的另外一个起因。我们都清楚，对于大多数应用软件，系统在初次运行之前要做一些初始化工作：

(1) 运行程序的绝对路径；

(2) 系统使用的各种组件的注册，如 ocx、ODBC 等；

(3) 系统运行的一些其他保存参数；

(4) 系统的注册控制。

所有这些工作，当然可以在安装完毕后通过手工方式进行设置，但是如果这样做可能会带来几个问题：

(1) 程序安装者必须对应用程序相当熟悉。这通常需要软件方派人现场安装，对于大型项目这是必不可少的，但是大多数小型应用软件的服务商不可能提供这种安装，试想如果 Microsoft Office 采用这种方式，那么 Microsoft 每天所做的事恐怕就是派人去做软件的安装服务了，没有几个人去做软件开发了。

(2) 系统的稳定性和安全性受到威胁。如果软件开发方把系统的初始化和注册方式公开，那么从软件本身来说，很可能遭到其它蓄意的破坏；另外从软件开发方的利益来说，很可能在这个过程中泄露一些系统内部的核心技术，无论如何这对于双方都无法接受的。

(3) 造成系统维护困难。系统难免会出现某些问题，往往会要求重新更新系统，这样必然要将该应用程序进行重新安装，当然这样又要请软件开发方派专人去做。这样一来不仅增加了用户方的系统维护费用，而且也加重了软件开发方的维护负担，对双方都没有什么好处

1.1.3　“打包”的工程学概念

软件是一个工程的概念，自然有一套有别与一般工程开发的独特流程，而且这种流程也因具体项目不同而存在一定的差异性，但是从根本上来说各种软件开发过程主要的环节还是一致的。作者以个人经验而言（可能与某些软件工程的教科书有一定出入），将软件开发过程概括如下。

1. 需求分析

本阶段的主要任务是开发方和用户相互交流，明确项目所要完成的最终目标是什么、采用的开发模式是什么、具体要求是什么、以及项目的可行性和成本效益分析等一系列分析。这一阶段是软件工程核心环节，往往耗时最长，难度也最大。

2. 概要设计

通过需求分析，开发方明确了项目具体要求和开发采用模型，然后针对这些进行系统分析和设计，即系统概要设计。具体进行以下几个工作：

(1) 根据要求和模型设计出相应的各种编程模型；
(2) 根据项目具体要求和建模，进行系统模块划分；
(3) 对系统各模块进行概要设计，形成设计文档和相关的数据字典；
(4) 相成输入输出信息流图和系统模块设计流程；
(5) 分配任务，进行相关人员的培训。

3. 软件设计

经过前一阶段的工作，可以说一个具体项目在逻辑上已经实现了，接下来的这一阶段就是把这种逻辑通过编码实现。如果前一阶段工作做得比较好，一般来说，软件设计阶段的问题不是很大。如果开发人员充足，本阶段一般只占开发周期的四分之一的时间。

当然，本阶段的核心问题就是程序的编码。除了注意编码的风格外，特别要强调的一点是在编码阶段尽可能去掉 Bug。由于现在软件开发一般采用集成开发环境（IDE），所以对模块编程的调试带来了很大的方便性。同时，在编码阶段要注意开发文档的制作，将开发文档制作分散到开发的各个阶段，不要将文档的制作集中在软件开发的后期完成。因为，把文档制作分散到个阶段，可以随时记录开发过程中遇到的各种问题以及相应的解决方案，如果把这一过程放到开发的最后进行，程序员通常会忘记一些东西，而这些东西往往是非常宝贵的经验，如积累下来，无论对程序员本身还是软件开发方都是一笔宝贵的“财富”。

4. 软件测试

软件测试主要包括模块和系统集成测试，其中模块测试主要放在软件设计阶段进行，本阶段主要针对系统集成的测试。测试的内容包括系统逻辑测试、异常测试、破坏性测试以及安装测试。对于测试中采用的方法很多，不过实际中最有意义的是临界值测试法，即找出各种逻辑分支的临界点，针对该点以及其左右领域的随机数据进行测试，这样做下来，如果没有出现什么问题，基本可以说软件不存在什么大的问题了。再经过其他几方面的系

统测试，便可宣告测试成功了。

在本阶段，安装测试主要是针对安装程序而进行的。具体来说，就是测试安装程序是否满足程序包的三个基本要求，如果满足了，从功能上来说安装程序通过测试了。

5. 交付与总结

经过软件测试，基本上可以说，应用程序可以投入到市场或交付给用户了。不过在正式交付之前，软件开发方必须有针对地对软件进行某种包装，以适应用户和市场各方面的需求。特别是对软件的安装程序应该做最后完全的制作，软件开发方应充分利用各种资源，进行安装程序媒体的制作（包括安装程序用户界面的最后完善以及生动的用户安装引导）。

当完成一项具体工程之后，包括安装制作策略在内所有全过程进行适时的总结，形成相应的开发文档和报告，并针对在软件开发后期用户方的反馈作出相关的分析和对策报告以备后用。

最后要做好相应的软件安装的技术支持。

通过前面的介绍，似乎安装程序只与软件的系统集成测试有关。但是实际情况并不是这样的。在软件设计全部过程都要涉及安装程序的设计，都要为将来应用软件产品的发行来考虑和设计、开发，具体软件开发各阶段应用程序和安装工程设计的任务如下：

(1) 从需求分析开始，软件开发方必须形成一个清晰的安装程序概念。项目的最终成果以什么样的方式、采用什么样的介质、通过什么样的手段安装到客户的目标计算机上，是采用分发式还是专人安装的方式，是客户申请式还是公司维护式，这一系列的问题必须在需求阶段确定下来，从这个角度我们又一次看到需求分析的重要性。

(2) 同样在概要设计阶段要为安装程序确定运行环境的支持文件，包括各种 dll，ocx，sys 等文件以及项目进行过程中独立开发的各种相应系统文件，并且确定这些文件在目标计算机的安装目录结构。另外在软件概要设计阶段的同时要设计将来安装程序所做的各种初始化工作以及相应的控制。

(3) 相对于前两阶段，软件设计阶段与安装程序没有直接关系，但是编码过程中应尽可能保证应用程序的健壮性，这将间接影响到安装程序的可靠性。同时在编码阶段的后期，将是安装程序的制作过程。

(4) 而在软件测试阶段，对安装程序的功能将进行全面测试，针对打包形成的程序包要满足的三个基本条件进行测试。

(5) 在最后一个阶段，在保证安装程序基本功能的基础上添加各种令人赏心悦目的安装界面以及多媒体，使应用软件真正成为一件可以交付的产品。

软件工程的各阶段以及相应的安装程序制作设计关系详细情况如图 1-1 所示。

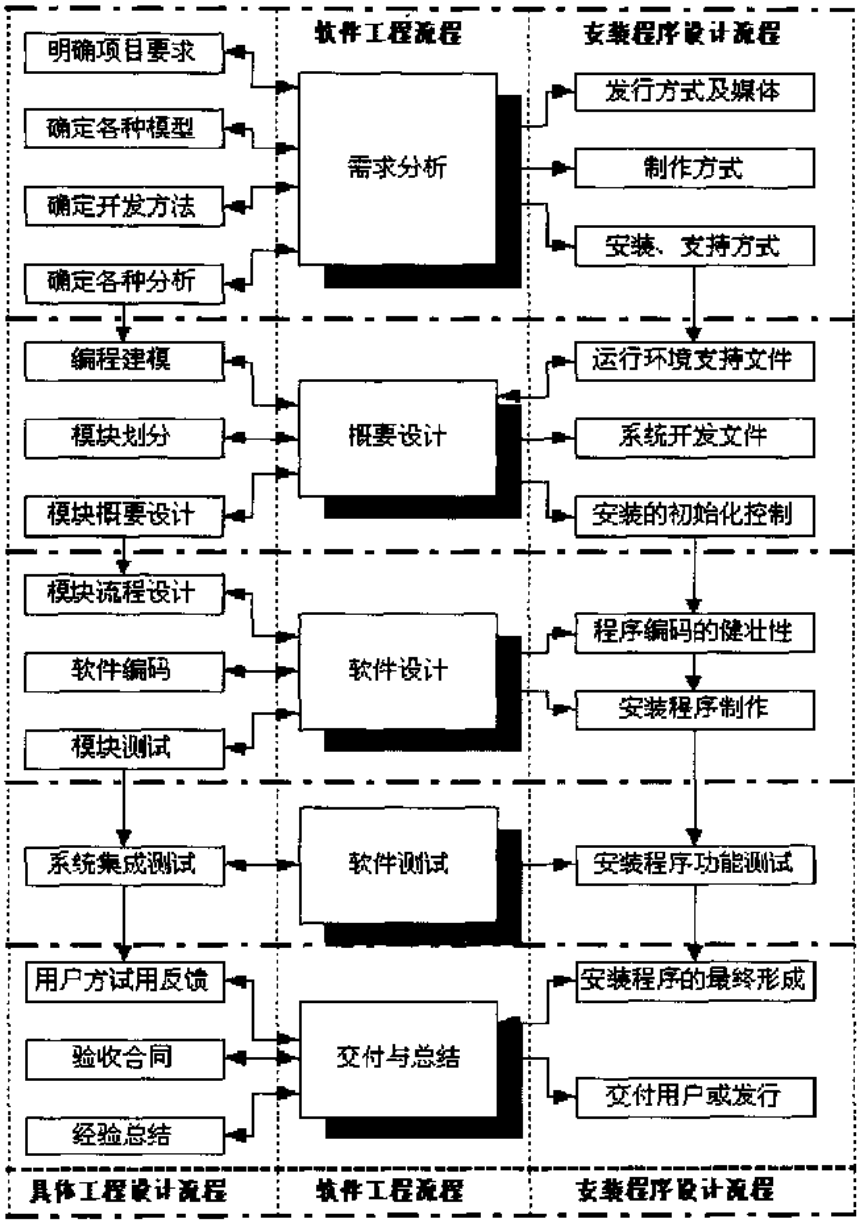

图 1-1　软件开发过程及相应项目、安装程序制作设计流程

1.1.4　“打包”的美学原理

既然“打包”的主要作用是将应用程序无缝地交付给用户，或者说是以某种产品的形式交付给用户，那么这种交付的方式是很有讲究的。安装程序经过软件测试以后，从功能上已经可以满足程序“包”的 3 个基本要求，也就意味着安装程序可以发行给用户了，但是如果这样制作出来的程序包投放到市场，销路一定会很好吗？当然不会。市场不允许这样的产品，同样软件开发方也不会这样做。试想，软件开发方开发某个产品，前后投入大量人力、物力，开发出来的产品的功能也很完善，决不会仅仅因为最后发行的安装程序

在功能以外其它方面的欠缺，而有损整个产品计划，也决不会放弃采用各种制作手段进行安装程序的最终完善。而这种完善主要集中在制作各种诱人的界面和多媒体安装画面上，其设计原则主要是根据人的认知心理趋向和认识美学的相关原理。所以一个好的软件开发方，不仅有大量经验丰富的系统分析员和程序员，而且拥有大批富有想象力的美工设计人员和市场分析人员。正是这些人的综合作用，才会推动软件的进步和公司的发展；也正是这个缘由，我们很有必要了解有关原则。

人的认知心理过程相当复杂，即使是心理学家至今还无法仍然完全把这种过程彻底弄清楚，只能根据人类的某些行为的长期观测和试验，作出这种认知心理的趋向性判断，也就形成了对软件有益的认知心理趋向。即人的认知趋向取决于认知的先验，而这种先验往往受到外界环境的强烈刺激作用。原理虽然很简单，但是真正能利用这一原理要花一些功夫的。如果我们把这一原理以程序化的思维进行处理，可得到下面的认知心理趋向流程图 1-2。

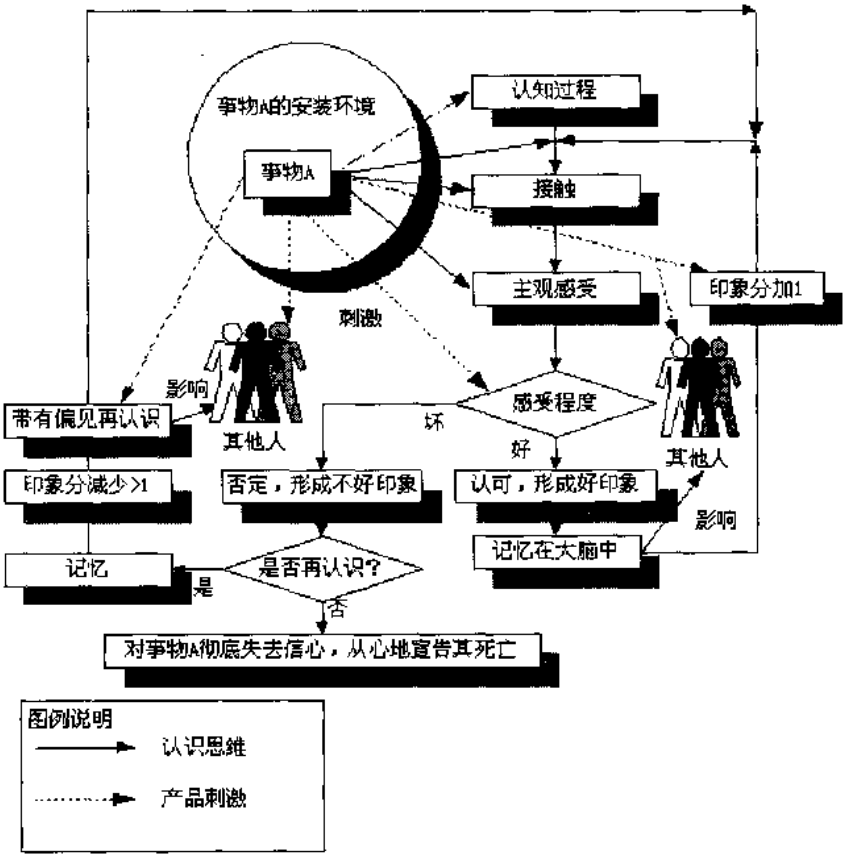

图 1-2　认知心理趋向程序化流程图

当然这个认知趋向流程作了一定程度的简化，虽然不能完全把整个过程模拟出来，但从框架以及大方向上基本体现了认知心理趋向的本质。通过图 1-2 可以看出，如果用户认知过程对事物 A 印象好，那么对相应事物再接触时，在没有具体接触之前，“印象分”已经增加了许多，这样如果形成一个良性无限循环，可想而知这是一个多么忠实的用户了，

即使软件开发方在其中可能出现某些产品失误，对整个过程也不会有什么影响（这是不可避免的）。相对而言，如果上述过程，用户的认知趋向走的是流程的另一分支，那么，从图 1.2 上清楚的看到，该用户对该产品的印象很差，甚至在心底宣告了其死亡，即使他在接触该产品，也总是带有某种挑剔或偏见去认识，除非该产品有重大的变化，否则该用户的再认识效果仍然不是很好。

那么怎样才能使用户尽可能朝好的分支去认识、接受产品呢？除了提高软件自身的质量和功能外，最重要的是加强认识过程中的各个环节的刺激。一般软件公司的刺激行为主要针对认知过程的前三个过程，即认知开始、接触以及主观感受。采取的方法也很多，诸如通过多种媒体进行宣传，制作精美的包装，采取各种各样的营销方案等，也起到了一定效果。但是软件开发方应该注意，以上提到的各种刺激手段是所有商业产品惯用的方法，我们应该看到软件产品与一般产品的不同之处，应该运用各种具有软件特色的刺激手段去刺激用户，争取客户，应该对整个认知心理趋向的各个环节都予以刺激，具体情况参见图 1.2 中虚线所示。

说到刺激，其实存在的方法也很多，但遵循的原则是一致的，那就是满足美学标准，说得更具体一些就是满足用户多方面的感官需求，特别针对刺激中的关键部件——安装程序的设计尤为重要。

1. 有针对性

在安装程序制作设计中，首先应该明确被“打包”软件针对的用户群是哪一部分。如果是针对普通个人用户，我们必须请各种美工设计人员以及音乐制作人员对产品的安装画面和方案做全面的剖析，特别是一些娱乐性的软件，诸如游戏、多媒体软件等，更是要求在这个方面下一番功夫，一定做到安装画面引人入胜、赏心悦目，给人一种美的享受，起到未正式使用软件之前，单单通过安装画面过程就吸引众多用户去接触该软件，产生一种“欲望”；而对于公司级的软件产品，就应该采用另外一套安装程序制作方法。有些软件开发方往往回走入这样的误区，不仅把应用程序制作得五光十色而且其安装程序也做得令人眼花缭乱。这样的做法，往往会引起公司使用的反感。一般而言，公司级的软件产品，从美学上来讲，应该给人一种非常大方、朴实的感觉，所以包括应用程序以及安装程序在内的各种开发制作都应该注意这个原则，而不应该过分去追求界面的丰富多彩。我们很少见过 Microsoft 以及 Oracle 等公司的产品出现令人眼花缭乱的界面和安装过程，然而给人的感觉却很好，让用户用起来特别舒服。

2. 其次，要明确美学中的配合原则

我们都有这样的经验，几种单独看起来的东西很好看，但是组合起来就显得不是那么和谐了，往往还不如单一事物所产生的效果，这就是美学中的配合原则。在安装程序制作过程中，我们不应该为追求某种效果而破坏整体安装效果。例如，在 Windows 环境下各种对话框的底色均为浅灰色，如果你想在安装过程中的某个环节弹出一个红色底色的对话框，使用该安装程序的用户一定会被吓一跳，的确给人印象“深刻”，但这是一种不良的刺激，通常会把整体安装程序效果破坏了。所以在安装程序制作过程中至少要满足以下美学配合原则：

- 尽可能与 Windows 的标准靠近；
- 界面的整体和局部尽可能不要形成强烈反差；
- 安装界面色彩的使用不要过分追求极端，应以素淡、大方为主；
- 界面尺寸以及显示位置的匹配一定要符合突出重点、操作方便、接近习惯的几点基本要求；
- 声音与色彩的配合要符合安装过程的内容；
- 图像的配合要和谐；
- 安装过程的各个环节的连接要连贯、自然，不要形成一种跳跃式的过程；
- 人机界面要友好，尽可能避免跳出不必要的对话框；
- 提示语言要简明扼要，符合使用者的习惯。

3. 要明确安装程序制作全部过程中的美学标准

安装程序不仅在安装界面上给人以美的感觉，而且在安装制作工程实施的整个过程都要贯彻美的标准。我们不仅要制作精美的安装程序，而且还要为用户提供精美、详实的安装知道手册以及用户使用手册等一系列的用户文档，同时还要有针对地提供相应的安装以及其它的技术支持。

1.2　用户真正的要求

开发软件，从某种意义上就是满足用户各种各样的要求，而正是这种不断发展、更新的要求推动了软件产业的迅速发展。如同软件的多样性和差别性，用户的要求也是千差万别的，而且这种要求的多样性会随时间以及周围环境、技术等一系列因素而变化，可以毫不夸张地说用户的要求永无止尽。当然任何事物都不是绝对的，在一定的环境和条件下，用户的这种要求可以定性化、条件性地满足。所以，我们可以运用各种手段去认识、发掘、满足用户的各种要求。

为了真正弄清楚用户的要求，需要软件开发方针对不同用户群的要求做到心中有数。即使不能完全掌握用户的全部要求，至少应该做到“有底”，特别是对于系统分析员和项目负责人，平时一定要积累这方面的经验。因为如果这部分工作做得比较好，可以减少在具体项目实施过程中的需求分析的时间，所以广大高级程序员应该注意一下。

当然如果单纯讨论有关软件需求的问题，那么恐怕可以写成一本厚厚的书，由于本书主要介绍安装程序制作，所以针对安装程序的具体情况，简单介绍一下用户对安装程序的要求。

1.2.1　用户基本要求

1. 可执行性

作为一个安装程序，从某种意义上来说是一种应用程序外壳，其主要功能是无缝地、友好地把应用程序安装到终端用户的目标计算机上，并且经安装后的应用程序可以完成软件任务书中规定的各项功能，同时在一定程度上保证系统的自维护性和可更新性。这也是