

软件工程技术丛书

HZ BOOKS

软件复用与构件技术系列

软件复用技术

在系统开发 过程中考虑复用

Software Reuse Techniques

Adding Reuse
to the Systems Development Process

(美) Carma McClure 著 廖泰安 宋志远 沈升源 译

PH
PTR



机械工业出版社
China Machine Press



中信出版社
CITIC PUBLISHING HOUSE

软件工程技术丛书

TP311.5
39

软件复用与构件技术系列

软件复用技术

在系统开发 过程中考虑复用

Software Reuse Techniques
Adding Reuse
to the Systems Development Process

北方工业大学图书馆



00529331

(美) Carma McClure 著 廖泰安 宋志远 沈升源 译



机械工业出版社
China Machine Press



中信出版社
CITIC PUBLISHING HOUSE

本书是关于复用技术的详细指南,从复用概念、复用管理技术、应用者复用技术、生产者复用技术几个方面,一步步地教导读者如何将复用加入到软件开发过程中,并获得复用的最大效益。本书不仅说明了如何复用源码,而且还讲解了如何复用需求说明、设计、测试脚本、项目计划、文档、对象框架等,并通过案例研究详细介绍了16种不同的复用技术。

本书将复用原理与技术应用很好地结合在一起,适用于有软件开发背景并且想学习如何实施复用驱动软件开发的专业人士,也可以作为高等院校相关专业的软件工程课程的参考用书。

Authorized translation from the English language edition entitled Software Reuse Techniques: Adding Reuse to the Systems Development Process by Carma McClure, published by Pearson Education, Inc, publishing as Prentice Hall PTR, Copyright © 1997 Carma McClure.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanic, including photocopying, recording, or by any information storage retrieval system, without permission of Pearson Education, Inc.

Chinese simplified language edition published by China Machine Press.

Copyright © 2003 by China Machine Press.

本书中文简体字版由美国Pearson Education培生教育出版集团授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号: 图字: 01-2002-0640

图书在版编目(CIP)数据

软件复用技术: 在系统开发过程中考虑复用 / (美) 麦克劳 (McClure, C.) 著; 廖泰安等译. -北京: 机械工业出版社, 2003.2

(软件工程技术丛书. 软件复用与构件技术系列)

书名原文: Software Reuse Techniques: Adding Reuse to the Systems Development Process

ISBN 7-111-11576-7

I. 软… II. ① 麦… ② 廖… III. 软件工程 IV. TP311.5

中国版本图书馆CIP数据核字(2003)第004412号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑: 王冰飞

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003年2月第1版第1次印刷

787mm×1092mm 1/16·16.25印张

印数: 0 001 - 5 000册

定价: 35.00元

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

译者序

正如本书作者所指出的，复用和程序设计本身具有同样长的历史。程序设计中的宏、过程、函数、子程序，以及对象类和软件包等都是软件复用的形式。近年来，软件复用作为一种新技术之所以越来越引起人们的重视，是因为人们认识到要想提高软件的生产效率，解决软件危机问题，软件复用是一个重要途径。软件复用的潜力还远远没有发挥出来，软件不仅可以在代码级复用，而且可以在系统的需求分析、软件设计、编码、测试以及形成文档等各个开发阶段和各个层次上实现复用。因此有必要将复用融入到软件开发生命周期过程中，使复用从单项技术或开发人员的自发行为，变成系统技术和有组织的开发过程。本书从软件开发方法学的角度介绍了在软件生命周期过程中如何考虑复用的具体规程。本书虽然写于1997年，在科技高速发展，特别是计算机技术日新月异的今天，它没有包含数据挖掘、知识管理等最新技术，但其介绍的基本内容，特别是结合实践对技术规程的具体说明，对复用的实施仍然有很强的指导作用。

软件是知识的表达。大家一定不会忘记牛顿的名言，“如果我比别人看得更远些，那是因为我站在巨人们的肩上”。人们正是通过世代的继承和积累，才不断丰富人类的知识宝库，软件复用归根到底是知识的复用，它不但符合人们的认识规律，而且作为人脑的辅助工具必将大大促进知识的积累和发展的过程。

由于译者水平的限制，译文中难免有错误和不妥之处，欢迎广大读者批评指正。

前言

软件浪费

“不浪费则不会匮乏”历来不是软件开发者的座右铭。相反，人们提倡将在大多数软件项目中的浪费作为同类系统开发哲学的一部分。传统的软件开发把从头开始建立每个新系统看作为用户/客户服务的最好方法，而不注意和关心“软件的重复创建”出现的次数。

不必要的重复创建和在大多数软件项目中出现的资源的巨大浪费，一直没有引起人们充分的重视。无数软件项目耗费大量资源去开发的软件构件和已经存在的那些软件构件非常相似。软件的浪费是普遍存在的，因为不同的软件系统看上去相似多于差异。对软件系统的比较分析表明，通常，在多个系统中，系统功能的60%到70%是相同的。

除非进行特殊的评估，否则无论使用何种技术或建立何种应用系统，软件浪费都将继续出现。最近出现的新技术，例如软件自动化、面向对象技术和客户/服务器技术都没有改变这个事实。然而，这些新技术具有更容易从软件相似性中获得效益的优点。一些软件的相似性被预先定义并建立在软件工具中（例如在代码生成器中可复用代码的模式）；另一些被建立为可复用构件，存放在软件复用库中，从而可用于开发新系统。

因为一般的软件系统是由相似的部分组成的，所以每个新的软件系统的大部分能够并将由预先定义的可复用构件组装而成。尽管软件复用常常被忽略，但它确实是解决过去几十年的软件危机问题的最佳答案。

可复用构件

可复用构件可能是程序源代码，但复用更大的效益常常来自于更广泛和更高层的复用。软件需求说明、设计、测试脚本、项目计划、文档、对象框架和子程序都是可复用构件类型的例子。通常，可复用构件将产生任何可交付的软件项目。

在可复用构件类型列表中最重要的是可复用大纲，它能辅助生成各类项目文档、报告和计划。因为这些大纲能以字处理模板形式提供，所以能够快速和容易地建立它们，并且可以在复用目录中得到。其效益是不仅节省了时间，而且确保了公司标准的执行和保持了跨软件开发过程和系统的一致性。这使系统具有更高的质量，它们更易理解、维护和复用。除此之外，这类可复用构件不会影响系统开发者的创作欲望，因为文档制作不是他们乐意做的事情。这些大纲中的许多部分，可能已经存在于开发组织的某些部门中并能由所有软件开发组应用。

其他两类软件构件在可复用构件价值列表中也是重要的，它们是设计构件和测试构件。

设计构件是重要的，因为最大程度地利用复用的最好地方是在设计时期。设计构件是比编码更高级的抽象，它们不涉及特定的实现，所以更易移植和更易复用。而且，因为设计是比编

码更昂贵的开发任务，复用设计可能比复用编码更能节约成本。除此之外，如果在设计和编码实现之间有某些可追溯性的话，在设计级的复用可以促成在编码级的复用。可追溯性可以由CASE工具自动地处理。关注设计构件可以解决在用不同程序设计语言编写的构件之间的差异和不相容等问题。

测试构件，诸如测试数据、测试脚本、测试用例和测试计划，也被认为是非常重要的。在某些情况下，尽管由于语言和工具不同，代码不能被复用，但测试构件可能是可复用的。复用测试构件能确保系统具有更高的测试覆盖率，其结果是得到更可靠、更可维护的系统。因为测试和设计一样是非常昂贵的任务，复用测试构件能节省大量的项目时间和资源。

复用效益

抓住复用机会能使软件生产率、质量和成本得到巨大改善。运用复用能得到的主要效益是：

- 提高软件生产率
- 缩短软件开发时间
- 以更少的人员去开发软件
- 人员、工具和方法更易于从一个项目转移到另一个项目
- 降低软件成本
- 生产更高质量的软件
- 改善软件系统的交互性
- 提供竞争优势

复用障碍

尽管复用的思想很简单，但它的实现却并不简单。今天，大多数的美国公司还没有规范的复用程序。尽管复用有和程序设计本身同样长的历史，它是保留着被认可的正确实施的好想法的软件技术，但是，和大多数好的意向一样，复用还没有真正地付诸实施。

多数软件开发者和他们的管理层关于复用的态度是，如果他们知道如何以高性价比、低风险和适宜的形式实现它，他们将实施复用。问题是他们使用的大多数软件方法学不包括复用。这些方法学没有明确的定义何时、何地 and 如何作为开发过程的一部分实施复用。进而，他们既不理解项目团队、用户和管理层的复用角色和职责，也不理解支持复用需要的工具。

本书的目的

本书是关于复用的。然而，它不是软件复用定义和度量概念的泛泛讨论，而是实际开展复用工作的详细说明。本书的目的是填补如何实现软件复用细节方面的空白。它是一本详细指南，一步步教导读者将复用加入到软件开发过程中（包括面向对象和传统结构化/非面向对象的过程

两者), 并获得复用能提供的最大效益。

本书是为任何有软件开发背景并且想学习如何实施复用驱动软件开发的专业人员准备的。

- 系统开发者和维护者能使用本书学习复用如何改变和简化创建和维护软件系统的工作, 以及如何实施复用特有的技术。
- 软件方法学专家担负着选择和定义所在公司软件生命周期方法学和软件标准的重任, 他们能使用本书了解支持复用的软件生命周期方法具有哪些特征。通过了解, 他们将能选择为复用提供最好支持的方法学并能扩展他们公司现行的方法以支持复用。
- 数据库/数据仓库管理组能以本书为指南, 建立一个具有分类模式的复用库/复用目录, 以及建立可复用构件的验证过程。
- 软件项目管理者能使用本书学习如何将复用分解为项目调度、资源需求、软件生产率和质量管理, 以及人力管理等因素。
- 软件培训者和教育者能使用本书为软件管理者和技术人员制定软件复用课程的内容。

最后, 正在打算进行复用的起步工作的公司能使用本书定义和实现规范的、特定的复用程序, 它覆盖了软件组、商务单位, 也许最终是整个企业。

本书的结构

本书分为4个基本部分:

- (1) 复用综述
- (2) 复用管理技术
- (3) 应用者复用技术
- (4) 生产者复用技术

综述包括三章。第1章是概述, 用一些公司复用的成功案例介绍了复用的情况。这些案例不仅说明了这些公司如何实施复用, 而且为读者提供了有价值的复用教训和实际实施复用的宝贵建议。第1章的目的是引起读者学习更多关于如何应用复用的兴趣和激励公司认真考虑采用复用软件开发模式。

第2章开始将读者的注意力引入复用的细节。在这一章中, 探讨了复用如何改变软件过程的一般问题。向读者说明复用思想和复用任务影响软件过程的地方。例如, 在战略系统规划中加入战略复用计划, 在系统设计评审和检查中加入复用检查。

读者将明白遵循复用模式意味着软件生命周期过程必须以复用开始, 以复用结束, 并且在中间的所有阶段包含复用。本章说明了三个特性, 软件过程必须使这三个特性支持复用。这些特性是:

- (1) 影响软件开发做出判断的复用主题。
- (2) 拓宽开发的视野, 从一次一个系统到多次一组相关系统所需的复用需求。
- (3) 复用视图, 将软件开发过程分为应用者和生产者复用两个方面。

第3章继续说明复用如何改变软件生命周期的过程, 但是将读者引向细节。复用专门的技术填补了如何实施复用细节的空白。因为复用是一种不同的软件模式, 复用的实施者必须掌握这

些复用技术。在本书中，总共定义了16种复用技术。

本书将复用技术分为三个范畴：

- (1) 管理技术，它们在公司复用程序级和系统开发项目级上辅助管理复用的实施。
- (2) 应用者技术，它们在如何使用可复用构件构造软件系统和他们相关的交付方面辅助开发者。
- (3) 生产者技术，它们辅助识别、建立和包装用于复用的软件构件。

除非读者理解在软件过程环境中的何处应用复用技术，否则复用技术的价值很有限。在第3章中给出了每种复用技术的简单介绍，并说明了每种技术适用于软件生命周期的哪个阶段。

本书的第二、三、四部分包含本书的主体和如何实施复用的主要内容。它包含了16章，其中的每一章都对一种复用技术进行了详细的、一步步的规程说明。本书最后的术语表指明了在“复用范畴”中这些术语如何被定义。例如，术语“领域”在复用技术中就有和在数据库技术中不同的含义。

每项技术综合介绍了业界复用专家思想和经验。包括在每章结尾的参考文献为读者提供了这些专家发表的文章。在本书中介绍的许多技术已被工业界测试过，它们已由一些公司在实际中应用。只要可能的话，关于如何使用该技术的实例都将包含在技术说明中。

本书的结束语提醒读者正确地开展工作以实现复用，因为复用能获得效益，这些努力是值得的。

致谢

我将感谢Lucent Technologies，俄勒冈州的Blue Cross/Blue Shield，DST Systems公司，GTE Telephone Operations，Sprint和UPS公司，他们帮助我发展和应用了本书中介绍的一些复用思想和技术，并且允许我采访他们的软件管理者和员工，以学习他们复用的进取精神。

我还想感谢在1996摩根镇软件复用研究会中我工作组中的成员，他们帮助我发展了在第2章中介绍的软件过程的概念。感谢新西兰惠灵顿市维多利亚大学的Robert Biddle、Fidelity Investments的Cyrille Cobe、JP Morgan的Darrell Fernandes、Texas Instruments的Ronnie Hardaway和D'Ann Hunt、西弗吉尼亚大学的Rich Kurnik、Asset的Gitika Magoon-Dalla、Reuse公司的Ruben Prieto-Diaz、Prologic公司的Jay Reddy、Nortel的Don Seto、以前在DST Systems公司工作的Steve Shortt、Lockheed的Frank Svoboda和Texas Instruments的Kevin Terry。

特别要感谢Jeanne Follman、Mary Trimble、Kay Paine和Leissa Shahrak，感谢他们的评论和贡献。

最后，我想感谢Extended Intelligence公司的总裁William McClure，是他鼓励我写了这本书。

Carma McClure

McClure博士的e-mail地址是：73733.2037@compuserve

McClure博士的网址是：<http://www.reusability.com>

目 录

译者序

前言

第一部分 复用综述

第1章 复用：一种软件模式3

- 1.1 复用与对象技术的密切关系3
- 1.2 复用的经验5
 - 1.2.1 延续到第二个系统5
 - 1.2.2 越大越好5
 - 1.2.3 对复用进行投资6
 - 1.2.4 复用不是偶然的6
 - 1.2.5 非正式的复用6
 - 1.2.6 复用技术是一门学科7
 - 1.2.7 从业务处理开始实施复用8
 - 1.2.8 复用驱动的软件开发过程8
 - 1.2.9 复用的代价9
 - 1.2.10 类库管理11
 - 1.2.11 几个好的超大类11
 - 1.2.12 在领域级别内尝试复用11
 - 1.2.13 复用的重组12
- 1.3 如何成功地实施复用12
- 1.4 参考文献13

第2章 在过程中复用14

- 2.1 复用驱动的方法是目前缺少的关键部分14
- 2.2 复用是如何改变软件过程的15
 - 2.2.1 复用主题15
 - 2.2.2 复用要求16
 - 2.2.3 复用视图17
- 2.3 渐进式地改变19
- 2.4 复用在哪些地方适合于软件过程19
- 2.5 扩展战略系统规划并包含复用20
- 2.6 扩展系统生命周期并包含复用22

- 2.6.1 首先应该作为一个复用应用者22
- 2.6.2 应用者复用微型生命周期22
- 2.6.3 复用驱动的项目评审23
- 2.6.4 复用项目计划24
- 2.6.5 最终的复用评价24
- 2.7 在系统生命周期中引入生产者复用25
 - 2.7.1 领域工程27
 - 2.7.2 将领域工程置于软件过程中29
- 2.8 考查复用驱动软件过程32
- 2.9 总结32
- 2.10 参考文献33

第3章 复用技术综述34

- 3.1 在软件方法中加入复用技术34
- 3.2 管理技术34
 - 3.2.1 公司级复用管理技术34
 - 3.2.2 项目级复用管理技术36
- 3.3 应用者复用技术37
- 3.4 生产者复用技术38
- 3.5 复用技术介绍40

第二部分 复用管理技术

第4章 复用就绪评估45

- 4.1 概述45
- 4.2 目的45
- 4.3 基本原理45
- 4.4 关键问题45
- 4.5 推荐的工具46
- 4.6 实例46
- 4.7 规程49
- 4.8 参考文献70

第5章 建立公司复用计划72

- 5.1 概述72
- 5.2 目的72

5.3 基本原理	72	9.6 实例	117
5.4 关键问题	72	9.7 规程	117
5.5 推荐的工具	73	9.8 参考文献	122
5.6 实例	73	第10章 项目复用评估	124
5.7 规程	75	10.1 概述	124
5.8 参考文献	84	10.2 目的	124
第6章 复用的组织	85	10.3 基本原理	124
6.1 概述	85	10.4 关键问题	125
6.2 目的	85	10.5 推荐的工具	125
6.3 基本原理	85	10.6 规程	125
6.4 关键问题	85	10.7 参考文献	129
6.5 实例	86	第11章 复用成本/效益跟踪	130
6.6 规程	87	11.1 概述	130
6.7 参考文献	98	11.2 目的	130
第7章 推广复用	99	11.3 基本原理	130
7.1 概述	99	11.4 关键问题	130
7.2 目的	99	11.5 推荐的工具	131
7.3 基本原理	99	11.6 规程	131
7.4 关键问题	99	11.7 参考文献	135
7.5 推荐的工具	100	第三部分 应用者复用技术	
7.6 实例	100	第12章 选择应用包	139
7.7 规程	103	12.1 概述	139
7.8 参考文献	108	12.2 目的	139
第8章 建立项目复用计划	110	12.3 原理	139
8.1 概述	110	12.4 关键问题	139
8.2 目的	110	12.5 推荐的工具	139
8.3 基本原理	110	12.6 实例	140
8.4 关键问题	110	12.7 规程	141
8.5 推荐工具	111	12.8 参考文献	145
8.6 规程	111	第13章 选择可复用构件	146
8.7 参考文献	115	13.1 概述	146
第9章 复用成本/效益分析	116	13.2 目的	146
9.1 概述	116	13.3 原理	146
9.2 目的	116	13.4 关键问题	146
9.3 基本原理	116	13.5 推荐的工具	146
9.4 关键问题	116	13.6 规程	147
9.5 推荐的工具	117		

13.7 参考文献	155
第14章 冗余检查	156
14.1 概述	156
14.2 目的	156
14.3 原理	156
14.4 关键问题	156
14.5 推荐的工具	157
14.6 实例	157
14.7 规程	159
14.8 参考文献	162
第15章 标识候选的可复用构件	163
15.1 概述	163
15.2 目的	163
15.3 原理	163
15.4 关键问题	163
15.5 推荐的工具	164
15.6 实例	164
15.7 规程	166
15.8 参考文献	170

第四部分 生产者复用技术

第16章 建立复用库	173
16.1 概述	173
16.2 目的	173
16.3 基本原理	173
16.4 关键问题	174
16.5 推荐的工具	174
16.6 实例	174

16.7 规程	176
16.8 参考文献	190
第17章 配置管理	191
17.1 概述	191
17.2 目的	191
17.3 基本原理	191
17.4 关键问题	192
17.5 实例	192
17.6 规程	193
第18章 建立可复用构件	200
18.1 概述	200
18.2 目的	200
18.3 基本原理	201
18.4 关键问题	201
18.5 实例	201
18.6 规程	202
18.7 参考文献	212
第19章 领域分析	215
19.1 概述	215
19.2 目的	216
19.3 基本原理	217
19.4 关键问题	218
19.5 推荐的工具	219
19.6 实例	220
19.7 规程	224
19.8 参考文献	231
术语表	234
后记	243

第一部分

复用综述

第 1 章

复用：一种软件模式

1.1 复用与对象技术的密切关系

设想一下，如果出现一种仅谈论对象技术而不提及复用（*reuse*）的情形，那么这将是很少见的。复用经常被看作为从对象技术中所寻求的最大益处之一。例如，如图1-1所示，在最近一次对象技术讨论会上，出席者们将缩短产品上市时间并将复用作为采用对象技术的头等原因¹。对于软件组织来讲，因为软件工业的发展趋势是逐步地采用对象技术作为其主要的软件开发方法，所以它们也在逐步地提高对复用的兴趣。许多组织已经选择使用面向对象的方法作为它们的开发方法，因为这能使它们取得更多的复用。毫无疑问，这些组织如果想成功地利用对象技术，就必须掌握复用。

1995年，在Boston Object World会议中的调查结果

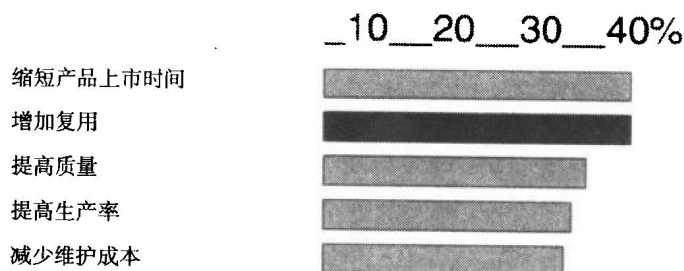


图1-1 采用对象技术的原因

软件复用（*software reuse*）是一种由预先构造好的、为复用目的而设计的软件构件来建立或者组装软件系统的过程。它的基本思想是非常简单的，即放弃那种原始的、一切从头开始的软件开发方式，而利用复用，由公共的可复用构件来组装新的系统，这些可复用构件包括对象类（*object class*）、框架（*framework*）或者软件体系结构等。表1-1列出了不同类型可复用构件的例子。

表1-1 不同类型可复用构件的例子

在位于不同抽象层次、不同大小的软件组成部分或者构件，以及文档交付方面，不同类型可复用构件的例子包括：
— 应用包
— 子系统
— 数据类型定义
— 设计（例如，像分解视图等设计模型）
— 规格说明（例如，商务模型）

-
- 代码
 - 文档
 - 测试用例和测试数据
- 在面向对象开发中，不同类型可复用构件的例子包括：
- 应用框架（例如，一般化的设计和代码框架）
 - 用例（例如，利用用户/外部代理在对话框中表示系统要完成的一组相关的动作序列）
 - 商务对象类（例如，高层对象类）
 - 分析和设计模型（例如，对象模型、系统操作模型、对象交互图、类的继承层次结构）
 - 类定义（例如，类内部结构的描述和该类在继承树中的位置）
 - 基本对象类（例如，底层的类，像日期类和字符串类等）
 - 类库（例如，一组支持某一领域的相关类的组合，像图形用户界面和数据库等）
 - 方法（例如，类的服务或者类的行为）
 - 测试包（例如，测试用例、测试数据和预期结果）
 - 函数（例如，程序模块）
 - 系统体系结构（例如，高层设计）
 - 文档（例如，分析文档和设计文档）
 - 项目、测试和实施计划的框架（例如，建立项目计划的格式等）
-

复用的潜力是巨大的，因为每一个新应用系统的大部分都可以由可复用构件组装而成，但需要的一个前提是上述可复用构件能够被预先确定、建立，并且容易被系统开发人员获取。

来自得克萨斯州Irving的GTE电话运营公司认为，复用的潜力是非常巨大的。它们计划在三至五年的时间内，将使其应用系统的80%至90%由库中可复用的对象构件组装而成²。

对象技术许诺能按时地向客户交付节省成本的、高质量的和具有柔性的软件系统。然而，理解到这一点是十分重要的，即只有通过复用，对象技术才能够带来上述益处。另外，我们必须懂得使用对象技术并不会导致自动的复用。一个常见的误解是，如果我们使用了对象技术，那么就等于我们在实施复用。其实对象技术和复用是两码事，尽管对象技术鼓励复用这种思想，并且封装（*encapsulation*）和继承（*inheritance*）也支持复用。而且，对于复用来讲，对象技术既不是充分的也不是必要的。换句话说，对象技术并不是复用的前提条件。有许多没有使用对象技术，而能成功地实施复用的例子。

复用是一门独立的学科，它应被看作为对象技术的一种必要的补充。有些人认为对象技术对复用的依赖会超出问题的本质。引用GTE电话公司体系结构方面的副总裁助理的话就是，“没有复用的对象技术是一场最可怕的噩梦。如果在大的应用项目中不能得到大量复用的话，那么这这将是一场灾难”³。那些将复用看作为面向对象开发过程中所产生的副产品的人，对复用所取得的水平曾感到非常的失望。

惟有将上述两种强有力的技术结合起来，我们才能期望得到它们曾许诺过的巨大益处。问题是如何将二者结合起来？对一个公司来讲，在软件开发的过程中，为了从对象技术和复用的结合中得到真实的、明显的改善，需要做哪些事情？对那些曾尝试过复用的公司来讲，在实际情况中实施复用所带来的益处有很大的差异。有些公司曾报告在一个软件项目中取得了20%的复

用⁴，其他公司当在一系列相关的系统中实施复用时，得到了80%至95%的复用，并且充分地提高了生产率和质量⁵。

1.2 复用的经验

让我们来密切地关注下述那些谈论过它们复用经验的公司，学习一下要成功地实施复用需要做什么⁶。一家系统集成公司认为复用要想取得成功，需要好的、强有力的管理和遵循大量的规范。Blue Cross/Blue Shield公司认为，适当的设施和耐心能使复用得到成功。而UPS（联合包裹运送服务公司）认为，需要在项目级别之上实施复用。最后，Sprint公司认为重点应放在较大的和高层次的可复用构件上。

这些公司中任何一个复用成功的案例，都示范了如何从对象技术和复用所许诺的好处中获益。它们同时表明，复用会给软件开发过程带来变化，而这种变化和面向对象模式所要求的有很大的区别。而且，这些变化不仅适用于面向对象的开发，同样也适用于传统的非面向对象的开发。这些变化包括：

(1) 复用必须被看作是一种长期的策略，管理者应对它负责。

(2) 复用的实施必须跨越组织间的界限，扩展到不同的项目中。

(3) 复用必须得到组织结构的支持，该组织结构将系统开发人员的角色和责任分为可复用构件建立者和系统组装者。

1.2.1 延续到第二个系统

为了成功地实施复用，公司必须愿意在时间上和资金上对复用进行投资。同时，该公司也必须愿意等待从投资中所得到的回报。很少出现这种情况：在实施复用的第一个项目或者在复用程序的头几个月，就能从复用中得到回报。例如，美国弗吉尼亚州的一家系统集成公司在其第一个面向对象开发项目中，并没有从复用中得到任何的益处。直到第一个项目中所建立的系统体系结构和商务逻辑在第二个项目中被复用的时候，才得到了复用的益处。在第二个项目中，因为系统的70%是由可复用构件来建立的，所以该项目比计划提前6个月完工。

根据上述开发经验，该系统集成公司在复用方面的建议是必须处理两种复用要求：

(1) 应该教会系统开发人员如何使用预先确定的、通用的系统体系结构。

(2) 需要额外的时间来建立可复用构件，因为它们需要更多的文档，而且必须为复用的目的而设计。

1.2.2 越大越好

Sprint公司的复用经验支持这种观点。该公司的40个曾在Smalltalk项目上一起工作的开发人员建立了一个复杂的、大型的网络存货清单管理系统，该系统之所以复杂，是因为它涉及到不同的学科和商务领域。项目的复用目标是取得较大的复用和高层次的可复用构件。

项目复用的主要领域是框架。因为框架包含了大量的功能，所以从复用框架中得到的回报

也是非常大的。Sprint公司的框架主要采用了表示某些公共功能的抽象类形式，复用该框架的开发人员由抽象类来实现具体类。在建立应用程序时，框架可以用来缩减整个功能领域的范围，从而允许开发人员将工作重点放在应用中特有的功能上。Sprint公司所建立的框架用于完成日常需要的功能，例如，数据存取、安全和用户接口等功能。

在网络存货清单管理系统的开发过程中，项目团队成员应努力识别在当前应用程序和其他应用程序中公共的需求，然后开发处理这些需求的框架。根据一位项目团队成员的经历，他建议，“从事项目开发的人员必须识别出系统的共性，然后决定如何将这此共性抽象到较高的层次来形成框架。寻找当前工作的系统和未来系统的共同点，然后封装这些共同点，从而使其更容易被复用。为了保证框架将来的可复用性，要将其看作为一个软件产品（*software product*），必须为其建立完整的文档来说明当它被复用时，如何对其进行定制和子类化”。

因为对Sprint公司来讲，框架的概念是相当新的，所以它还没有成为公司范围内可供使用的可复用构件。公司还必须定义处理框架的正式机制，例如验证过程、培训和复用库等。Sprint公司建立了一个支持框架的专用业务组，用来传递可供复用的构件并收集新的框架。

Sprint公司的观点是正式地实施复用和度量复用带来的效益是需要花费时间的。特别地，对公司进行重组来支持复用是一个重要的问题。由于必须设计框架来支持复用，因此决定由谁来承担这个责任是任何一个认真对待复用的组织所必须回答的一个基本问题。

1.2.3 对复用进行投资

Blue Cross/Blue Shield是另一家认为为了从复用中得到回报，必须对其进行投资的公司。该公司的第一个为索赔领域开发的面向对象项目就定位于为未来的复用。在第一个项目期间，项目开发团队成员有意识地寻找具有复用潜力的构件，并且从复用的角度来对它们进行设计。

建立和测试复用类，使项目延期了两至三个月。该公司为索赔领域总共建立了69个通用类，期望它们中的30%将有企业级的复用潜力。同时，该公司认为在第一个项目上所花费的建立可复用类的额外时间是值得的，因为接下来的项目可以通过复用这些索赔类来产生高质量的系统，并且能节省大量的时间，而这些类是被包含在大多数保险应用程序中的核心类。

该公司还建立了一个特别的组织来支持可复用类和仓库的管理。

1.2.4 复用不是偶然的

与Blue Cross/Blue Shield公司一样，来自美国路易斯维尔的UPS公司也没有期望从第一个复用项目中获得益处，也不期望复用是偶然发生的，他们知道设计可复用构件必须是软件开发过程中一个明确的部分，必须集成过程、方法和工具，并且以一种有效地促进复用的方式来使用它们。

1.2.5 非正式的复用

新泽西州的一家系统集成公司发现，尽管在没有实施复用的情况下，可能从对象技术中得到某些益处，但二者结合以后会得到更多的益处。该公司将二者结合，以固定的成本、在规定