

- 汇集专业作者团队十余年经验心血
- 完美呈现软件测试精髓和热点技术
- 测试行业国内外知名专家鼎力推荐



软件测试 技术指南

斛嘉乙 符永蔚 樊映川 著
腾尚时代 组编



机械工业出版社
CHINA MACHINE PRESS

软件测试技术指南

腾尚时代 组编
斛嘉乙 符永蔚 樊映川 著



机械工业出版社

本书介绍了国内外先进的软件测试技术和测试理念：包含软件测试理论、软件质量、软件测试过程、软件测试方法、软件测试管理、软件测试工具以及 Web 项目测试和 APP 项目测试等。全书覆盖了基础和高阶的软件测试知识，并结合目前市场需求的岗位技能，提供了极具参考价值的测试实例。本书是腾尚时代软件职业培训学校全体老师精心打造的一本软件测试领域专业书籍，力求使更多的求职者和读者更好地学习软件测试的相关知识，并找到更理想的软件测试工作岗位。

本书适合于从事软件测试领域的技术人员及希望从事软件测试的其他专业人员阅读，也适合计算机、软件、自动化等相关专业的学生与老师参考。

图书在版编目 (CIP) 数据

软件测试技术指南/斛嘉乙，符永蔚，樊映川著. —北京：机械工业出版社，2019.1

ISBN 978-7-111-61475-3

I. ①软… II. ①斛… ②符… ③樊… III. ①软件-测试-指南
IV. ①TP311.55-62

中国版本图书馆 CIP 数据核字 (2019) 第 000624 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

策划编辑：尚 晨 责任编辑：尚 晨

责任校对：张艳霞 责任印制：孙 炜

天津千鹤文化传播有限公司印刷

2019 年 1 月第 1 版·第 1 次印刷

184mm×260mm·14.5 印张·346 千字

0001-3000 册

标准书号：ISBN 978-7-111-61475-3

定价：55.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

网络服务

服务咨询热线：010-88361066

机工官网：www.cmpbook.com

读者购书热线：010-68326294

机工官博：weibo.com/cmp1952

010-88379203

金书网：www.golden-book.com

封面无防伪标均为盗版

教育服务网：www.cmpedu.com

前 言

随着互联网的发展，软件的规模和复杂性都大幅度提升，用户对软件的要求也越来越多，除了软件的基本功能之外，软件的性能以及用户对隐私和敏感数据的安全性等方面用户也特别重视。

软件测试是软件整个研发过程中最重要的一个环节，同时也是软件质量保证最主要、最关键的手段之一，其理论知识和测试工具都在不断的革新。随着 IT 行业的不断发展，软件测试人才的需求也在不断增加，现软件测试工程师已经成为了 IT 行业中一个热门的职位，其主要负责如功能测试、性能测试、自动化测试以及安全测试等。

本书由浅入深的介绍测试各领域的专业知识，可达到快速入门，短期提升，全面掌握的阅读目的。其中包含测试基础理论的全面解说，测试管理的各类方式，测试领域当下最流行的测试工具使用，测试最常用的脚本语言的技能掌握，还有 Web 项目和 APP 项目的测试方法和测试思路的拓展。

对于刚开始学习软件测试的零基础同学，还是想在测试道路上进阶的初中级测试工程师，相信这本书都能给大家一个学习思路的引导，从而稳步提升测试技能。

本书由腾尚时代信息科技有限公司创始团队斛嘉乙、符永蔚、樊映川编写，书中凝聚十多年的测试领域研究成果，以通俗易懂，易学实用为导向，使阅读本书如同真实工作体验，便于更快理解测试该如何进行以及使用相应的专业技能。

限于作者的经验和水平有限，书中难免有不足之处，恳请广大读者批评指正。如有疑问和建议可以发邮件，作者电子信箱 8773465@qq.com。

编者

序言——测试之所见

当前大多数人把软件测试理解为功能测试，他们认为只要理解测试理论，测试流程，再熟练测试用例设计的方法就可以了。所以很多接触这个行业的人，在工作一两年之后，就想着往性能测试、自动化测试方面发展，结果是越学越迷茫，真正的原因是很多人对测试的理解还不够。那么该如何做呢？

下面简述一下作者的看法。

首先测试需要发挥其主动性。因为最终的产品是给用户使用的。也就是说，测试从项目立项开始就需要更全面理解每个需求的意义是什么，关注前期的需求分析和讨论，参与需求的评审，这样可以根据产品需求合理的设计测试方案以及安排好测试时间。

其次就是测试用例的设计。测试用例是用来指导测试工作，也是测试人员必须要完成的工作。测试用例的设计可以从三个维度（业务、用户、方法）来考虑，此外需要对测试用例进行管理，通常写完测试用例后要进行评审，后期在执行中还要不断的更新测试用例。

接下来关注测试流程，注重项目进度的把控。通常这些是项目测试经理来管理，其实作为一个合格的测试人员也要协助领导进行流程的管理，同时对缺陷要进行跟踪管理。特别是针对遗留缺陷的分析以及可能存在的风险。

最后在项目上线后，需要对项目进行相应的总结。总结整个项目过程中遇到的问题有那些，如流程方面、技术方面等。这些问题解决的办法是什么，后续有什么可以借鉴的方案或改进的措施。

除了上述的一些看法，做为一个合格的测试人员还需要具备以下几方面的能力。

沟通能力，其主要体现在两方面。一方面是思路，主要指业务和技能方面，比如在工作中不管是与开发人员沟通，与客户沟通，还是与领导沟通，在沟通前要首先整理好自己的思路。另一方面是表达，主要是指说话的技巧，其实体现的就是情商，比如在测试中发现 Bug 后，有些人会直接说：“这个地方有 Bug”，有些人则会委婉一点说：“这里执行时跟需求不一样，您过来看一下”。总之，不管是开发也好，还是测试也好，我们的工作目标是为了提供高质量的软件给用户，所以在实际工作中，需要尽最大可能理解对方，从而提高工作效率，也就是说要学会换位思考。

思维能力，其主要体现在两方面。一方面是逻辑思维，首先要清楚逻辑思维是一种确定的、有条理、有根据的思维；其次在工作中需要充分熟悉需求和业务，从专业的角度去思考问题。另一方面体现在发散思维，也就是探索性测试，比如在工作中可以天马行空的去设想用户的操作行为，使用最多的方法就是逆向思维。

学习能力，主要体现在两方面。一方面是技能方面。以一个 Web 系统来说，可能涉及的语言有 C++、Java、.Net、PHP、Javascript、HTML 等，还会涉及数据库服务器以及应用服务器等。从这点上看，就要求测试人员能够读懂多门语言。另一方面就是业务方面，从目前来看，软件几乎涉及所有的行业，包括银行、电商、物流以及政府的业务办理等，而软件

的发展也越来越人性化，所以要求测试人员不断学习和理解不同行业、不同业务的规范和标准。

团队合作能力，现今软件越来越复杂，一个高质量的项目需要不同部门，不同职位的人相互协同实现，仅凭个人能力是不可能完成的。目前很多公司特别注重团队意识，经常不定期组织一些团队扩展活动，来增强团队之间的合作意识。作为测试工程师必需具备高度的团队合作精神，为保证软件产品的质量做出贡献。

最后，做为一名测试人员，还需要有责任心、信心、耐心。平台不同、定位不同，人生的价值就会截然不同，请做好您的职业规划。

时间一直在不断的做减法——且行且珍惜。

斛嘉乙

2018年9月于家中

目 录

前言

序言

第 1 章 软件测试理论	1
1.1 软件概念	1
1.1.1 软件发展史	1
1.1.2 软件生命周期	3
1.1.3 软件体系结构	4
1.2 软件研发模型	5
1.2.1 瀑布模型	5
1.2.2 快速原型模型	6
1.2.3 螺旋模型	7
1.2.4 RUP 流程	8
1.2.5 敏捷模型	10
1.3 软件测试基本概念	11
1.3.1 软件测试发展	11
1.3.2 软件测试目的	12
1.3.3 软件测试原则	13
1.4 软件测试模型	14
1.4.1 V 模型	14
1.4.2 W 模型	15
1.4.3 H 模型	16
1.4.4 X 模型	16
1.4.5 敏捷测试	17
1.5 软件缺陷	17
1.5.1 软件为什么会引入缺陷	17
1.5.2 缺陷种类	18
1.6 测试用例	18
1.6.1 测试用例的重要性	19
1.6.2 测试用例写作思路	19
第 2 章 软件质量	21
2.1 质量	21
2.1.1 质量铁三角	21
2.1.2 软件质量	22

2.2	质量管理体系	22
2.2.1	ISO	23
2.2.2	CMM/CMMI	24
2.2.3	6Sigma	27
2.3	软件质量特性	28
2.3.1	功能性	29
2.3.2	可靠性	29
2.3.3	易用性	29
2.3.4	效率	29
2.3.5	可维护性	30
2.3.6	可移植性	30
2.4	软件质量活动	30
2.4.1	SQA 由来	30
2.4.2	SQA 工作职责	31
2.4.3	软件度量	32
第3章	软件测试过程	34
3.1	测试阶段划分	34
3.2	单元测试	34
3.2.1	单元测试环境	34
3.2.2	单元测试策略	35
3.2.3	单元测试常见的错误	36
3.2.4	单元测试工具	37
3.3	集成测试	37
3.3.1	集成测试环境	37
3.3.2	集成测试策略	37
3.3.3	集成测试分析	39
3.3.4	集成测试工具	39
3.4	系统测试	40
3.4.1	系统测试环境	40
3.4.2	系统测试策略	40
3.5	验收测试	45
3.5.1	UAT 测试	45
3.5.2	α 测试	45
3.5.3	β 测试	45
3.6	回归测试	46
3.6.1	回归测试流程	46
3.6.2	回归测试策略	46
3.7	软件测试的流程	47
3.7.1	测试角色与职责	47

3.7.2	测试计划与控制	47
3.7.3	测试分析与设计	48
3.7.4	测试实现与执行	49
3.7.5	测试评估与报告	50
3.7.6	测试结束活动	50
第4章	软件测试方法	52
4.1	测试方法划分	52
4.2	白盒测试	52
4.2.1	白盒测试常用技术	53
4.2.2	基本路径测试	55
4.2.3	白盒测试的优缺点	56
4.3	黑盒测试	56
4.3.1	等价类划分法	57
4.3.2	边界值分析法	60
4.3.3	判定表分析法	62
4.3.4	因果图分析法	63
4.3.5	正交试验法	66
4.3.6	流程分析法	68
4.3.7	状态迁移法	70
4.3.8	异常分析法	71
4.3.9	错误推测法	71
4.3.10	黑盒测试的优缺点	72
4.4	灰盒测试	72
4.5	静态测试	72
4.5.1	同行评审	73
4.5.2	阶段评审	75
4.5.3	同行评审与阶段评审的区别	75
4.6	动态测试	76
第5章	软件测试管理	77
5.1	配置管理	77
5.1.1	配置管理角色与职责	77
5.1.2	配置管理的流程	78
5.1.3	配置管理工具介绍	79
5.2	需求管理	80
5.2.1	什么是需求	80
5.2.2	需求的类型	80
5.2.3	需求工程	81
5.2.4	需求开发	81
5.2.5	需求管理	83

5.3	缺陷管理	85
5.3.1	软件缺陷的状态	85
5.3.2	缺陷管理的角色	86
5.3.3	缺陷管理基本流程	86
5.3.4	缺陷的等级划分	87
5.3.5	缺陷报告的内容	87
5.3.6	缺陷分析	88
5.4	风险管理	89
5.4.1	风险管理的过程	89
5.4.2	项目中常见的风险	90
5.5	敏捷风险管理	91
5.5.1	敏捷项目的理解	91
5.5.2	敏捷项目需求的管理	93
5.5.3	敏捷项目时间的管理	93
5.5.4	敏捷项目成本的管理	94
5.5.5	敏捷项目质量的管理	94
5.5.6	敏捷项目沟通的管理	96
5.5.7	敏捷项目风险的管理	97
第6章	测试工具的介绍	98
6.1	配置管理工具——SVN	98
6.1.1	VisualSVN 的安裝配置	98
6.1.2	VisualSVN 服务端的使用	101
6.1.3	TortoiseSVN 的安裝配置	107
6.1.4	TortoiseSVN 客户端的使用	109
6.2	缺陷管理工具	115
6.3	性能测试工具	116
6.3.1	性能测试概念	117
6.3.2	性能测试指标	118
6.3.3	性能测试流程	120
6.3.4	Loadrunner 工具介绍	120
6.3.5	VuGen 录制脚本	122
6.3.6	Controller 场景设计	130
6.3.7	Load Generator 负载生成	137
6.3.8	Controller 场景运行	141
6.3.9	Analysis 结果分析	141
6.4	接口测试工具	143
6.4.1	接口测试	143
6.4.2	Postman	145
6.4.3	Jmeter	147

6.5	抓包工具	158
第7章	常用协议简介	161
7.1	OSI 参考模型	161
7.2	TCP/IP 协议	162
7.3	HTTP 协议	166
7.3.1	HTTP 协议特点	166
7.3.2	HTTP 连接方式	167
7.3.3	HTTP 报文结构	168
7.3.4	HTTP 请求	169
7.3.5	HTTP 应答	171
7.3.6	HTTP 通信机制	172
7.3.7	HTTP 缓存机制	173
7.4	HTTPS 协议	175
7.4.1	HTTPS 的特点	176
7.4.2	工作原理	176
7.4.3	SSL 协议	176
7.4.4	TLS 协议	177
第8章	Web 项目测试	178
8.1	Web 基础	178
8.1.1	Web 的发展	178
8.1.2	Web 的特点	179
8.1.3	Web 工作原理	179
8.1.4	Web 页面加载过程	180
8.2	Web 测试技术	181
8.2.1	Web 功能测试	181
8.2.2	Web 性能测试	183
8.2.3	Web 安全测试	185
8.2.4	Web 兼容测试	187
8.2.5	Web 接口测试	187
第9章	APP 项目测试	189
9.1	APP 发展	189
9.2	APP 测试流程	189
9.3	APP 测试方法	190
9.3.1	Android 与 IOS	190
9.3.2	UI 测试	190
9.3.3	功能测试	191
9.3.4	性能测试	192
9.3.5	安全测试	193
9.3.6	兼容性测试	193

9.3.7	接口测试	194
9.3.8	用户体验测试	194
9.4	APP 测试工具	194
9.4.1	ADB	195
9.4.2	AAPT	202
9.4.3	Monkey	203
9.4.4	DDMS	210
9.4.5	Fiddler	212
9.5	APP 测试与 Web 测试的区别	217
9.6	H5 页面测试	218
参考文献		219

第1章 软件测试理论

本章主要介绍软件的发展、软件生命周期以及较为流行的研发模型、软件缺陷等理论知识，从而引出软件测试活动的基础理论：测试目的、测试原则、测试模型、测试用例等，为后续学习内容作好准备。

学习目标：

- 熟悉软件的生命周期
- 熟悉常见的软件研发模型
- 掌握软件缺陷以及等级的划分
- 掌握软件测试的模型：V模型、W模型、H模型、X模型
- 掌握软件测试的目的以及软件测试原则
- 掌握测试用例的写作思路

1.1 软件概念

软件（Software）是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲，软件被划分为系统软件、应用软件和介于这两者之间的中间件。在国标中对软件的定义是与计算机系统操作有关的计算机程序、规程、规则以及可能有的文件、文档及数据等。

由此可见，软件并不只是包括可以在计算机（这里的计算机是指广义的计算机）上运行的电脑程序，与这些电脑程序相关的文档一般也被认为是软件的一部分。简单地说软件就是程序加文档的集合体。

1.1.1 软件发展史

软件的发展大致经历了如下五个阶段：

1. 第一阶段（1946年—1953年）

该阶段属于汇编时代，软件是用机器语言编写的，机器语言是内置在计算机电路中的指令，由0和1组成（二进制数字）。因此，只有少数专业人员能够为计算机编写程序，这就大大限制了计算机的推广和使用。

由于程序最终在计算机上执行时采用的都是机器语言，所以需要一种称为汇编器的翻译程序，把用汇编语言编写的程序翻译成机器代码。编写汇编器的程序员简化了程序设计，是最初的系统程序员。

2. 第二阶段（1954年—1964年）

该阶段软件开始使用高级程序设计语言简称高级语言（与之对应机器语言和汇编语言被称为低级语言）编写，高级语言的指令形式类似于自然语言和数学语言，不仅容易学习，

方便编程，也提高了程序的可读性。

在1964年 Dartmouth 学院的凯梅尼 (John Kemeny) 和卡茨 (Thomas Kurtz) 发明了 BASIC (Beginner's All-purpose Symbolic Instruction Code) 语言。高级语言的出现催生了在多台计算机上运行同一个程序的模式，每种高级语言都有配套的翻译程序 (称为编译器)，编译器可以把高级语言编写的语句翻译成等价的机器指令。系统程序员的角色变得更加明显，系统程序员编写诸如编译器这样的辅助工具，使用这些工具编写应用程序的人，称为应用程序员。

在汇编和编译时期，由于程序规模小，程序编写起来比较容易，也没有系统化的方法，对软件的开发过程更没有进行任何管理。这种个体化的软件开发环境使得软件设计往往只是在人们头脑中隐含进行的一个模糊过程，除了程序清单之外，没有其他文档资料。

3. 第三阶段 (1965 年—1970 年)

该阶段处于结构化程序设计理论，由于用集成电路取代了晶体管，处理器的运算速度得到了大幅度的提高。因此需要编写一种程序，使所有计算机资源处于计算机的控制中，这种程序就是操作系统。

1967 年，塞缪尔 (A. L. Samuel) 发明了第一个下棋程序，开始了人工智能的研究。1968 年荷兰计算机科学家狄杰斯特拉 (Edsger W. Dijkstra) 发表了论文《GOTO 语句的害处》，指出调试和修改程序的困难与程序中包含 GOTO 语句的数量成正比，从此，各种结构化程序设计理念逐渐确立起来。

20 世纪 60 年代以来，计算机用于管理的数据规模更为庞大，应用越来越广泛，同时，用户对多种应用、多种语言互相覆盖的共享数据集合的要求越来越强烈。为解决多用户、多应用共享数据的需求，使数据为尽可能多的应用程序服务，出现了数据库技术以及统一管理数据的软件系统——数据库管理系统 DBMS (Database Management System)。

随着计算机应用的日益普及，软件数量急剧膨胀，在计算机软件的开发和维护过程中出现了一系列严重问题，许多程序的个体化特性使得它们最终成为不可维护的，“软件危机”就这样开始出现了。1968 年，北大西洋公约组织的计算机科学家在联邦德国召开国际会议，讨论软件危机问题，在这次会议上正式提出并使用了“软件工程”这个名词。

4. 第四阶段 (1971 年—1989 年)

该阶段属于结构化程序时代，20 世纪 70 年代出现了结构化程序设计技术，Pascal 语言和 Modula-2 语言都是采用结构化程序设计规则制定的，BASIC 这种为第三代计算机设计的语言也被升级为具有结构化的版本。此外，在 1973 年，美国贝尔实验室的丹尼斯·里奇 (D. M. Ritchie) 设计出了一种新的语言，这就是灵活且功能强大的 C 语言。

此外，IBM PC 开发的 PC-DOS 和为兼容机开发的 MS-DOS 都成了微型计算机的标准操作系统，更好用、更强大的操作系统被开发了出来。Macintosh 机的操作系统引入了鼠标的概念和点击式的图形界面，彻底改变了人机交互的方式。

20 世纪 80 年代，随着微电子和数字化声像技术的发展，在计算机应用程序中开始使用图像、声音等多媒体信息，出现了多媒体计算机。多媒体技术的发展使计算机的应用进入了一个新阶段。

这个时期出现了多用途的应用程序，这些应用程序面向没有任何计算机经验的用户。典型的应用程序是电子制表软件、文字处理软件和数据库管理软件。Lotus1-2-3 是第一个商

用电子制表软件，WordPerfect 是第一个商用文字处理软件，dBase III 是第一个实用的数据库管理软件。

5. 第五阶段（1990 年—至今）

该阶段软件中有三个著名事件：在计算机软件业具有主导地位的 Microsoft 公司的崛起、面向对象的程序设计方法的出现以及万维网（World Wide Web）的普及。

（1）在这个时期，Microsoft 公司的 Windows 操作系统在 PC 机市场占有显著优势，尽管 WordPerfect 仍在继续改进，但 Microsoft 公司的 Word 成了最常用的文字处理软件。20 世纪 90 年代中期，Microsoft 公司将文字处理软件 Word、电子制表软件 Excel、数据库管理软件 Access 和其他应用程序绑定在一个程序包中，称为办公自动化软件。

（2）面向对象的程序设计方法最早是在 20 世纪 70 年代开始使用的，当时主要是用在 Smalltalk 语言中。20 世纪 90 年代，面向对象的程序设计逐步代替了结构化程序设计，成为目前最流行的程序设计技术。面向对象程序设计尤其适用于规模较大、具有高度交互性、反映现实世界中动态内容的应用程序。其中 Java、C++、C#等都是面向对象程序设计语言。

（3）1990 年，英国研究员提姆·柏纳李（Tim Berners-Lee）创建了一个全球 Internet 文档中心，并创建了一套技术规则和创建格式化文档的 HTML 语言，以及能让用户访问全世界站点上信息的浏览器，此时的浏览器还很不成熟，只能显示文本。

软件体系结构从集中式的主机模式转变为分布式的客户端/服务器模式（C/S，Client/Server 的缩写）或浏览器/服务器模式（B/S，Browser/Server 的缩写），专家系统和人工智能软件从实验室走出来进入了实际应用，完善的系统软件、丰富的系统开发工具和商品化的应用程序的大量出现以及通信技术和计算机网络的飞速发展，使得计算机进入了一个大发展的阶段。

1.1.2 软件生命周期

生命周期（Life Cycle）的概念应用很广泛，简单说就是指一个对象的“生老病死”。对一个软件产品或软件系统而言也需要经历同样阶段，一般称为软件生命周期。软件生命周期大致分为六个阶段：如图 1-1 所示。

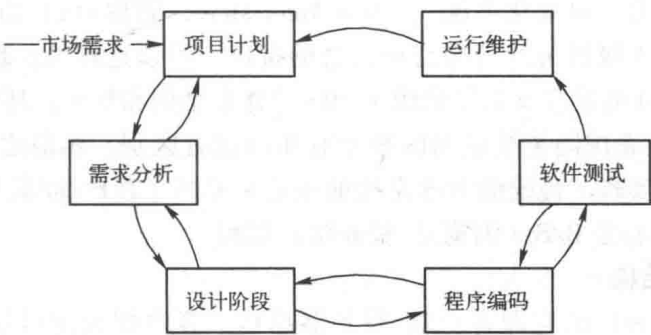


图 1-1 软件的生命周期

1. 项目计划阶段

此阶段主要是确定软件开发的总体目标，通过市场调研并给出功能、性能、接口等方面的设想以及项目的可行性分析，同时对项目开发使用的资源、成本、进度做出评估，制定项

目实施的计划（项目一级计划）。

2. 需求分析阶段

需求俗称软件的主体，所以需求分析阶段作为一个非常重要的阶段，它由需求分析人员和用户共同对软件需要实现的各个功能进行详细的分析并给予确切的描述，并编写软件需求说明书（Software Requirements Specification，简称 SRS）。

3. 软件设计阶段

该阶段俗称软件的核心，主要是由系统分析组（架构师和系统分析人员）根据需求分析的结果，对整个软件进行系统架构的设计，编写概要设计说明书（High Level Design，简称 HLD）。接下来由数据库设计员和开发人员根据需求说明书和概要设计说明书进行系统数据库设计以及编写详细设计说明书（Low Level Design，简称 LLD）。

4. 程序编码阶段

把软件设计的结果转换为计算机可运行的程序代码，使用 RDBMS 工具建立数据库。程序编码必须符合标准和编码规范，以保证程序的可读性、易维护性，保证程序运行的效率。

5. 软件测试阶段

此阶段主要是测试人员来检测软件是否符合客户的需求，是否达到质量的要求。一般在软件设计完成后，项目开发人员构建测试版本，以便测试团队进行测试，整个测试过程大致分为：单元测试、集成测试、系统测试、验收测试。

6. 运行与维护阶段

此阶段是软件生命周期中最长的阶段。在软件开发完成并正式投入使用后，可能有很多原因需要对软件进行修改，如软件错误、系统升级、增加功能、提高性能等。

1.1.3 软件体系结构

近年来，随着计算机技术与网络技术突飞猛进的发展，现代企业遇到了巨大的机遇与挑战，为了最大限度地利用现代计算机及网络通信技术加强企业的信息管理，很多企业建立了管理信息系统（Management Information System，简称 MIS）。一个完整的 MIS 应包括：辅助决策系统（Aided Decision Making System，简称 ADMS）、工业控制系统（Industrial Control System，简称 ICS）、办公自动化系统（Office Automation，简称 OA）以及数据库、模型库、方法库、知识库和与上级机关及外界交换信息的接口。可以这样说，现代企业 MIS 不能没有 Internet，但 Internet 的建立又必须依赖于 MIS 的体系结构和软硬件环境。

基于 Web 的 MIS 系统同传统的 MIS 技术有相似也有区别。相似之处在于技术的理念；区别之处在于技术的实现。传统的 MIS 系统的核心是 C/S（客户端/服务器）结构，而基于 Web 的 MIS 系统的核心是 B/S（浏览器/服务器）结构。

1. 什么是 C/S 结构

C/S（Client/Server）结构即客户端/服务器结构。客户端通常可以理解为安装在 PC、手机终端设备上的软件，是软件系统体系结构的一种。比如：QQ 软件、手机 APP 等，C/S 模式简单地讲就是基于企业内部网络的应用系统。与 B/S（Browser/Server，浏览器/服务器）模式相比，C/S 模式的应用系统最大的好处是不依赖企业外网环境，即无论企业是否能够上网，都不影响应用。

还有一类通信的软件，比如：百度云盘、迅雷下载等，也属于 C/S 结构。但是它们之

间的数据传输不需要经过服务器处理业务，可以直接通过客户端进行传输，这种结构通常称为 P2P (Peer to Peer) 点对点结构。

P2P 是可以简单的定义成通过直接交换来共享计算机资源和服务，而对等计算模型应用层形成的网络通常称为对等网络。对等网络，即对等计算机网络，是一种在对等者 (Peer) 之间分配任务和工作负载的分布式应用架构，是对等计算模型在应用层形成的一种组网或网络形式。

2. 什么是 B/S 结构

B/S (Browser/Server) 结构即浏览器/服务器结构。随着 Internet 技术的兴起，对 C/S 结构的一种变化或者改进的结构。在这种结构下，用户工作界面是通过浏览器来实现，极少部分事务逻辑在前端 (Browser) 实现，但是主要事务逻辑在服务器端 (Server) 实现，形成所谓三层 3-tier 结构。B/S 是目前互联网中应用最为广泛的系统结构。B/S 结构比起 C/S 结构有着很大的优越性，传统的 MIS 系统依赖于专门的操作环境，这意味着操作者的活动空间受到极大限制；而 B/S 结构则不需要专门的操作环境，在任何地方，只要能上网，就能够操作 MIS 系统，这其中的优劣差别是不言而喻的。

基于 Web 的 MIS 系统，弥补了传统 MIS 系统的不足，充分体现了现代网络时代的特点。随着网络技术的高速发展，因特网必将成为人类社会新的技术基石。基于 Web 的 MIS 系统必将成为网络时代的新一代管理信息系统，前景极为乐观。

3. 什么是 A/S 结构

A/S (Application Serving) 体系结构。A/S 体系结构通过设置应用服务器，将关键性的业务软件集中安装并进行发布，客户端可完全在服务器上执行所需的应用。A/S 结构利用 ICA 协议，将应用程序的逻辑从用户界面中分离开来，使得网络传输数据量很小，对网络带宽的要求低，平均每个用户仅占用 10K 左右，即使是通过电话线连接到 Internet，也能保证多个用户同时工作，提供数据的实时访问和更新。

另外，应用服务器与后台数据库通常采用局域网连接，计算和查询所需的大量数据都是基于 LAN 传输，因此远程用户的网络性能非常理想。同时在 A/S 结构中，网络中传输的仅是通过用户界面以及操作动作更新信息，因此系统的安全性更好。

1.2 软件研发模型

软件研发模型 (Software Development Model) 是指软件开发全部过程、活动和任务的结构框架。合理使用研发模型可以提供软件开发效率，降低研发成本，提升软件质量。

常见的研发模型包括需求、设计、编码和测试等阶段，有时也包括维护阶段。软件开发模型能清晰、直观地表达软件开发全过程，明确规定了要完成的主要活动和任务，用来作为软件项目工作的基础。

目前比较流行的研发模型主要有：瀑布模型、快速原型模型、螺旋模型、RUP 流程和敏捷模型。

1.2.1 瀑布模型

1970 年，温斯顿·罗伊斯 (Winston Royce) 提出了著名的瀑布模型。如图 1-2 所示。