

软件工程专业核心课程系列教材

软件工程专业英语

吕云翔 编著

清华大学出版社

清华大学出版社数字出版网站

WQBook  书文
局泉
www.wqbook.com

ISBN 978-7-302-36361-3



9 787302 363613 >

定价：39.50元

软件工程专业核心课程系列教材

软件工程专业英语

吕云翔 编著

清华大学出版社
北京

内 容 简 介

本书是按照最新《大学英语教学大纲》对专业英语的要求,以一个大学本科二年级学生 Kevin 与他的同学在一个酒店管理信息系统的实际项目中进行专业实践、直至经过求职面试进入一个 IT 企业工作为线索,将 IT 行业中所需的英语“听、说、读、写、译”基本技能与项目从开始到结束的整个流程有机融合起来,同时涉及云计算、移动互联网、大数据、物联网、社交网络、基于位置的服务、通用计算图形处理器、比特币、三维打印和谷歌眼镜等相关知识。

本书包括 10 个单元,每个单元都分为听与说、读与译以及写作部分。听与说部分描述了软件开发的技术场景;读与译部分给出了软件工程及 IT 相关的文章;写作部分则重点介绍如何撰写技术/商务文档和技术报告等。

本书注重听、说、读、写、译能力的全面发展,适用于高等院校软件工程及其相关专业、软件学院、各类职业信息技术学院和专业培训机构等。本书配有教辅资源,包括听与说录音、为教师提供授课 PPT,需要的教师可登录清华大学出版社网站 www.tup.com.cn 下载。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

软件工程专业英语 / 吕云翔编著. —北京:清华大学出版社, 2014

软件工程专业核心课程系列教材

ISBN 978-7-302-36361-3

I. ①软… II. ①吕… III. ①软件工程—英语—高等学校—教材 IV. ①H31

中国版本图书馆 CIP 数据核字(2014)第 099041 号

责任编辑:魏江江 赵晓宁

封面设计:常雪影

责任校对:梁 毅

责任印制:何 芊

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座

邮 编:100084

社总机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

课 件 下 载: <http://www.tup.com.cn>, 010-62795954

印 刷 者:北京富博印刷有限公司

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185mm×260mm

印 张:23

字 数:570 千字

版 次:2014 年 12 月第 1 版

印 次:2014 年 12 月第 1 次印刷

印 数:1~2000

定 价:39.50 元



前 言

英语是全球 IT 行业的行业语言, 英语技能是 IT 行业最基本的技能之一, 因此熟练掌握相关英语技能对于发展职业生涯具有积极的影响。

本书是按照最新《大学英语教学大纲》对专业英语的要求, 为开设软件工程英语课程而编写的教材。本教材在满足软件工程英语教学的同时, 注重学生的实际应用与调动学生的学习兴趣。本书选材广泛, 内容丰富, 涉及了软件开发的项目启动、需求获取、项目计划、团队合作、系统设计、系统实现、系统测试和系统交付的完整流程, 以及包括如何进行面试和在 IT 企业工作的场景。为了拓展知识, 还包括了云计算、移动互联网、大数据、物联网、社交网络、基于位置的服务、通用计算图形处理器、比特币、三维打印和谷歌眼镜等深刻影响着我们生活的信息技术。

本教材设计以一个大学本科二年级学生 Kevin 与他的同学在一个酒店管理信息系统的实际项目中进行专业实践, 直到经过求职面试进入一个 IT 企业工作, 成为这个企业的软件从业人员为线索来组织内容, 包含在 IT 行业中所需掌握的基本英语听说读写译技能, 涉及从接受项目开始到项目开发完毕这样一个完整的软件开发流程中 IT 相关人员所需的英语听说读写译技能。

本教材共有 10 个单元, 每个单元的训练都分为听与说、读与译、写作等方面。听力部分概要讲述与软件工程相关的知识, 对话部分涉及实际工作中与同学、客户或同事之间的交流; 读与译部分包括与软件工程相关的文章和与 IT 相关的最新文章; 写作部分讲解如何撰写软件工程文档以及商务文档等。本教材注重英语听说读写译能力的全面发展, 并与软件工程的专业课程紧密结合, 采用场景式教学和体验式学习相结合的方式, 实用性强。

另外, 本书配有教辅资源, 包括听与说录音、授课 PPT, 需要的教师可登录清华大学出版社网站 www.tup.com.cn 下载。使用本教材时, 建议教学时长为 36~48 学时, 教师可根据具体情况进行裁剪。

本教材适用于高等院校软件工程、计算机科学及其相关专业、软件学院、各类职业信息技术学院和专业培训机构等。教材中设计的听力、口语、阅读、写作和翻译的场景融合了角色扮演、多人对话、小

组讨论等行之有效的训练方法，能较好地满足课堂教学的需要，有利于学生在课堂上的即时消化吸收。

本教材在编写过程中得到了美国专家 Eric Langager 和 Law Phew 的指导，以及杨雪的大力帮助，在此表示衷心的感谢。

本教材试图融合听、说、读、写、译各项技能训练，书中难免会有不尽如人意之处，敬请专家与读者不吝赐教，以使该书臻于完善。编者联系方式 E-mail: yunxianglu@hotmail.com。

编 者

2014 年 6 月于北京

教学建议

本教材的教学建议如下：

对话部分 (Dialogue)：教师可先让学生听对话录音，并以提问的方式，引导学生根据所听信息概括对话主要内容，让学生了解和学习对话中涉及的相关知识。然后，教师可将学生分成三人小组，让其中一组或两组（分别）朗读这个对话，并纠正学生的发音；可让一组或两组参照已有对话并通过替换右边栏中的语句，组织完成一个类似的对话，并对学生完成的情况加以点评。

短文听力理解部分 (Listening Comprehension)：教师可先让学生听短文录音和短文后的问题，让学生根据所听内容选择正确的答案。若播放一遍短文学生感觉有难度，教师可酌情增加录音播放次数。教师最后公布答案，并且讲解相应的单词、短语、缩写及句子，解释这篇短文的重点和难点。另外，可让学生读一遍原文。

听写部分 (Dictation)：教师根据实际情况播放 1~3 遍短文录音，让学生根据所听内容填空，将短文补充完整。短文填充完整后，教师最后公布答案，并且讲解相应的单词、短语、缩写及句子，解释这篇短文的重点和难点。另外，可让学生读一遍原文。

阅读与翻译部分 (Reading & Translating)：Section A 部分的内容为软件工程领域知识，使读者深入了解和掌握软件工程相关专业知识。教师可让学生阅读文章（教师可根据文章的长短和难易程度来设定阅读的时间），并完成文章后的练习。之后教师公布练习答案，并讲解文章后的单词表、短语表、缩写词表和复杂句子来帮助学生进一步理解这篇文章。另外，教师最好还要讲解这篇文章所涉及的软件工程相关知识。Section B 部分的内容为 IT 领域相关知识，供读者开阔视野。教师可让学生阅读文章（教师可根据文章的长短和难易程度来设定阅读的时间），并完成文章后的练习。之后教师公布练习答案，并讲解文章后的单词表、短语表、缩写词表和复杂句子来帮助学生进一步理解文章。另外，教师最好还要讲解本篇文章所涉及的 IT 领域相关知识。如果课堂时间不够，可将 Section B 作为学生课后的作业。

写作部分 (Simulated Writing)：教师可先让学生阅读写作方法指导，并配合本教材的写作样例进行讲解和指导。教师还可根据实际情况设置场景，让学生根据写作指导并参照写作样例完成一篇类似文章。

如果课堂时间不够,教师可建议学生课下自学“写作部分”。

本教材各单元的教学内容、学习要点及教学要求以及课时安排参见下表。

教 学 内 容	学习要点及教学要求	课 时 安 排
Unit 1 Starting a Software Project	- 了解软件的基本概念、主要特点、组成要素; - 了解软件工程的基本概念、意义及其主要发展历程; - 了解云计算的基本概念以及其应用; - 掌握备忘录的写作方法	3
Unit 2 Capturing the Requirements	- 了解与用户沟通获取软件需求的过程; - 了解需求工程的基本概念、作用和主要目标; - 了解需求阶段的主要活动和主要方法; - 掌握需求分析的主要方法和最终产品; - 理解在软件项目中客户与最终用户的区别; - 了解移动互联网的基本概念以及其应用; - 掌握软件需求规格说明书的写作方法	4~6
Unit 3 Planning the Project	- 了解软件项目计划的基本概念、作用和主要目标; - 了解项目计划的主要活动和主要方法; - 了解软件项目中需要管理的变量要素及其之间的相互作用; - 了解大数据的基本概念以及其应用; - 掌握软件项目计划文档的写作方法	4~6
Unit 4 Working in a Team	- 了解软件项目中团队合作的重要性; - 了解软件项目中团队结构的分类、各自特点及适用情况; - 了解敏捷软件开发的基本概念和主要特点; - 了解物联网的基本概念以及其应用; - 掌握 PowerPoint 演讲稿的写作方法	3
Unit 5 Designing the System	- 了解软件系统设计的基本概念、作用和主要目标; - 了解软件设计阶段的主要活动和主要方法; - 了解用户体验和用户界面设计的重要性; - 了解社交网络的基本概念以及其应用; - 掌握软件设计规格说明书的写作方法	4~6
Unit 6 Implementing the System	- 了解系统编码实现的基本概念、作用和主要目标; - 了解编码阶段的主要活动; - 了解编写高质量代码的主要方法; - 了解程序设计语言的主要发展阶段、分类,及其典型代表语言; - 了解软件开发中的“20-80 法则”; - 了解基于位置的服务的基本概念以及其应用; - 掌握进度报告的写作方法	4~6

续表

教 学 内 容	学习要点及教学要求	课 时 安 排
Unit 7 Testing the System	- 了解软件系统测试的基本概念、作用和主要目标; - 了解软件测试阶段的主要活动和主要测试方法; - 了解冒烟测试的基本概念; - 了解通用计算图形处理器的基本概念以及其应用; - 掌握软件测试规格说明书的写作方法	4~6
Unit 8 Delivering the System	- 了解软件维护的基本概念、作用和主要目标; - 了解软件维护阶段的主要活动和主要方法; - 了解软件交付的主要工作; - 了解软件部署的主要工作; - 了解软件 Bug 和调试 Bug 的基础知识; - 了解比特币的基本概念以及其应用; - 掌握用户手册的写作方法	4~6
Unit 9 Taking an Interview	- 熟悉面试基本技巧和常见面试问题; - 了解软件配置管理的基础知识; - 了解计算机专业人员的主要职位及其主要工作; - 了解三维打印的基本概念以及其应用; - 掌握简历的写作方法	3
Unit 10 Beginning Your Work	- 了解 IT 公司的组织结构、管理层次、各部门职责; - 了解信息系统的主要分类; - 了解软件系统开发生命周期的主要阶段和各阶段主要任务; - 了解谷歌眼镜的基本概念以及其应用; - 掌握商务电子邮件的写作方法	3
教学总学时建议		36~48

说明:

① 本书为软件工程、计算机科学及相关本科专业“软件工程英语”课程教材,理论授课学时数为36~48学时,不同专业可根据不同的教学要求和计划教学学时对教材内容进行适当取舍。

② 非软件工程或计算机类本科专业使用本教材可适当降低教学要求。

③ 理论授课学时数36~48学时包含课堂讨论、练习等必要的课内教学环节。

④ 建议授课时间比例为听说部分50%、阅读部分30%、写作部分20%。

目 录

Unit 1 Starting a Software Project 启动软件项目	1
Part 1 Listening & Speaking	1
Dialogue: Starting a Software Project	1
Listening Comprehension: Software Engineering	3
Dictation: Mythical Man-Month & No Silver Bullet	4
Part 2 Reading & Translating	5
Section A: Software Engineering	5
Section B: Cloud Computing 云计算	9
Part 3 Simulated Writing: Memo	12
Unit 2 Capturing the Requirements 需求获取	16
Part 1 Listening & Speaking	16
Dialogue: Communication with Customers	16
Listening Comprehension: Software Requirements	18
Dictation: The Difference between Customer and End-User	19
Part 2 Reading & Translation	19
Section A: Software Requirements	19
Section B: Mobile Web 移动互联网	23
Part 3 Simulated Writing: Software Requirements Specification	27
Unit 3 Planning the Project 项目计划	63
Part 1 Listening & Speaking	63
Dialogue: Software Project Planning	63
Listening Comprehension: Software Project Planning	65
Dictation: Four Variables in Projects	66
Part 2 Reading & Translating	67
Section A: Software Project Plan	67
Section B: Big Data 大数据	71
Part 3 Simulated Writing: Software Project Plan	74

Unit 4 Working in a Team 团队合作	94
Part 1 Listening & Speaking	94
Dialogue: Team Structure	94
Listening Comprehension: Project Team	95
Dictation: Agile Software Development	96
Part 2 Reading & Translating	97
Section A: Team Structure	97
Section B: The Internet of Things 物联网	101
Part 3 Writing: PowerPoint Presentation	104
Unit 5 Designing the System 系统设计	109
Part 1 Listening & Speaking	109
Dialogue: Software Design	109
Listening Comprehension: Software Design	110
Dictation: User Interface Design	111
Part 2 Reading & Translating	112
Section A: Software Design	112
Section B: Social Networking 社交网络	117
Part 3 Simulated Writing: Software Design Specification	120
Unit 6 Implementing the System 系统实现	159
Part 1 Listening & Speaking	159
Dialogue: Creating High-Quality Code	159
Listening Comprehension: Writing the Code	161
Dictation: Concentrate on the Vital Few, Not the Trivial Many	162
Part 2 Reading & Translating	163
Section A: Computer Programming	163
Section B: Location-based Service 基于位置的服务	169
Part 3 Simulated Writing: Progress Report	172
Unit 7 Testing the System 系统测试	177
Part 1 Listening & Speaking	177
Dialogue: Software Testing	177
Listening Comprehension: Software Testing	178
Dictation: Smoke Test	179
Part 2 Reading & Translating	180
Section A: Software Testing	180
Section B: GPGPU 通用计算图形处理器	184

Part 3 Simulated Writing: Software Test Specification	188
Unit 8 Delivering the System 系统交付	201
Part 1 Listening & Speaking	201
Dialogue: Software Deployment	201
Listening Comprehension: Software Delivery	202
Dictation: Bug & Debugging	203
Part 2 Reading & Translating	205
Section A: Software Maintenance	205
Section B: Bitcoin 比特币	209
Part 3 Simulated Writing: User Guide	213
Unit 9 Taking an Interview 参加面试	220
Part 1 Listening & Speaking	220
Dialogue: Interview	220
Listening Comprehension: Expansion Company	222
Dictation: Software Configuration Management	223
Part 2 Reading and Translation	224
Section A: Careers for Computer Professionals	224
Section B: 3D Printing 三维打印	230
Part 3 Simulated Writing: Resume	233
Unit 10 Beginning Your Work 开始工作	238
Part 1 Listening & Speaking	238
Dialogue: Beginning Your Work	238
Listening Comprehension: The Organizational Structure of a Company	240
Dictation: Open Source Movement	241
Part 2 Reading and Translating	242
Section A: The Organizational Structure of a Company	242
Section B: Google Glass 谷歌眼镜	246
Part 3 Simulated Writing: Business E-mail	250
Glossary (词汇表)	254
Abbreviation (缩略语表)	274
Reference of Translation (参考译文)	278
Answers (练习答案)	337
Bibliography (参考文献)	353

Starting a Software Project

启动软件项目

Part 1 Listening & Speaking

➤ Dialogue: Starting a Software Project

(Kevin, Sharon, and Jason are three sophomores in the college of software in Beihang University. Today, they are attending a class meeting at the end of the fourth semester before starting the summer vacation.)

Teacher: Morning, everyone. In this vacation, you will implement a real project as your course project. There are some subjects you can choose in terms of your interests and experience. Please submit your decision to me within the next week.

Kevin: Excuse me, teacher. Is it a single task or can it be a cooperative work?

Teacher: Team work is recommended, because it benefits you to learn how to work together with your colleagues in the future and how to communicate, share, express, and understand ideas as a team member. But the size of the group should not be more than 4 persons.

Sharon: I'm interested in the subject of Four Seasons Hotel Management Information System, what about you, Kevin?

Kevin: Oh, it is my opinion too. And I think we can cooperate. Hi, Jason, would you like to join us? ^[1]

Jason: Oh, yes, I'd like to very much!

Sharon: Ok, now let's discuss on each person's responsibility.

Jason: Kevin is good at organizing and has lots of programming experience, so I think he can be our team leader or project manager, **in charge of** instructing our team and programming practice.

Sharon: I agree.

[1] Replace with:

1. Would you like to cooperate with us?
2. Would you like to collaborate with us?
3. Would you like to work together with us?

Kevin: Thanks for your trust. Ok, I will do my best. Besides coding, I think it is necessary to create a database and implement a suite of user interfaces for our software.

Jason: I am interested in databases and willing to be responsible for database building and management.

Sharon: I like art design, so I think I can do the UI design and document writing for our project.

Kevin: Oh! It seems this is a wonderful team and makes me very confident! Now, let's divide the work according to the phases of the project in general. As the team leader, I will be responsible for requirements, Jason will be in charge of design and Sharon will **take charge of** testing.

Jason: Next, we can **talk over** a rough prograss plan for our project.

Kevin: We can design, and then accomplish the UI operation according to the original requirements document provided by our teacher first. At the same time, Jason can be start building the database. Finally, we can accomplish coding together.

Sharon: It sounds wonderful. But I am afraid that the contents of the original requirements document will not be sufficient for our design, ^[2] first of all, we must do the requiremeats analysis based on the original requirements, and complete a formal Software Requirements Specification as our guidance of design.

Kevin: Oh, yes. Thanks for your important reminder. What do you think about it, Jason?

Jason: I agree with you completely.

(After meeting, Kevin asked for a document from the teacher about the hotel business requirements.)

Kevin: Hi, everybody. I have just got the business requirements of the hotel from our teacher.

Jason: Let me see. Oh, there is a list about their daily business and a table of related requirements. But it seems a little rough without enough detailed procedures, I am afraid.

Kevin: I see. And it does not mention the data flow and business model of this hotel.

Sharon: So, in that case, I think we need some communication with the customer (Four Seasons Hotel) to acquire more information.

Kevin: Yes. It's very necessary and I will call the customer to make an appointment with them. Before that, I think there is something we should do. That is, we had better do some homework to learn some knowledge about basic hotel business and management.

Sharon: That's right! It is very necessary to get some information about their business, and will be valuable for us to adequately and accurately understand the requirements.

Jason: Ok, I believe that the Internet can help us a lot.

Words

sophomore

[ˈsɒfəmoː] n. 大学二年级学生

rough

[rʌf] adj. 初步的, 粗略的

specification

[ˌspesɪfɪˈkeɪʃən] n. 说明书, 规范

reminder

[rɪˈmaɪndə] n. 提醒, 提示

Phrases

in charge of	负责, 领导
take charge of	担任, 监管
talk over	商议, 讨论

Abbreviations

UI	User Interface	用户界面
----	----------------	------

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ Listening Comprehension: Software Engineering

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which is correct about the development of software according to the article?
 - It emerged with software engineering at the same time.
 - For a half-century development, it has almost solved problems of high-quality, on-time and within-budget.
 - It was just a specialized problem solving and information analysis tool in its early years of development.
 - The laws which software evolves according to have changed absolutely during its development.
- Which point does not belong to the characteristics of software according to the article?
 - Easy to change the requirements
 - Easy to adapt the requirement changes
 - Difficult to measure the progress and process of creating
 - Difficult to test the correctness exhaustively
- Where was the phrase “software engineering” first used in 1968?
 - In a conference
 - In a thesis
 - In a journal

(D) In a magazine

Words

demonstrate

[ˈdemənstreɪt] v. 证明, 论证

concern

[kənˈsɜ:n] v. 担心, 忧虑

practice

[ˈpræktɪs] n. 实践, 实行

assimilate

[əˈsɪmɪleɪt] v. 吸收

intangible

[ɪnˈtændʒəbl] adj. 无形的

exhaustively

[ɪgˈzɔ:stɪvli] adv. 详尽地, 彻底地

address

[əˈdres] v. 处理, 满足

law

[lɔ:] n. 规则, 法则

framework

[ˈfreɪmwɜ:k] n. 构架, 体系结构

prototype

[ˈprəʊtətaɪp] n. 原型

discipline

[ˈdɪsɪplɪn] n. 学科

Abbreviations

NATO

North Atlantic Treaty Organization

北大西洋公约组织

➤ Dictation: Mythical Man-Month & No Silver Bullet

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

Frederick P. Brooks, Jr., is a Professor of Computer Science at the University of North Carolina at Chapel Hill. He is best ____ 1 ____ as the “father of the IBM System/360,” having served as ____ 2 ____ for its development and later as a manager of the ____ 3 ____ /360 software project during its design phase.

His book, **Mythical Man-Month**, is a most classic book on the ____ 4 ____ elements of software engineering. Since the first ____ 5 ____ in 1975, no software engineer's ____ 6 ____ has been complete without it. It was in this book that Brooks made the now-famous ____ 7 ____ : “Adding ____ 8 ____ to a late software project makes it ____ 9 ____ .” This has since come to be known as “Brooks's ____ 10 ____ .” Software tools and development ____ 11 ____ may have changed in the 30 years since the first edition of this book, but the **peculiarly** nonlinear economies of scale in ____ 12 ____ work and the nature of ____ 13 ____ and groups has not changed an **epsilon**.

In addition, Brooks is known for No Silver Bullet, which was ____ 14 ____ a 1986 IFIPS paper,

reprinted in 1987 in the IEEE Computer magazine and _____ 15 _____ in the second edition of The Mythical Man-Month later. Silver bullet is used to compare something to make software costs _____ 16 _____ as rapidly as computer hardware costs do. “No Silver Bullet” had wide _____ 17 _____ and proved **provocative**. It **predicted** that a decade would not see any _____ 18 _____ technique that would by itself bring an **order of magnitude** improvement in software _____ 19 _____. The decade has a year to _____ 20 _____; the author’s prediction seems safe. “No Silver Bulle” has stimulated more and more **spirited** discussion in the **literature** than has The Mythical Man-Month.

Words

mythical

['miθikəl] adj. 神话的, 虚构的

peculiarly

[pi'kju:liəli] adv. 特有地, 特别地

epsilon

['epsilən] n. 小的正数

provocative

[prə'vəkətiv] adj. 引起争论(议论, 兴趣等)的

predict

[pri'dikt] v. 预知, 预言

spirited

['spiritid] adj. 热烈的

literature

['litəritʃə] n. 著作, 文献

Phrases

order of magnitude

数量级

Abbreviations

IFIPS	International Federation of Information Processing Societies	国际信息处理学会联合会
IEEE	Institute of Electrical and Electronics Engineers	美国电气和电子工程师协会

Part 2 Reading & Translating

➤ Section A: Software Engineering

Virtually all countries now depend on complex computer-based systems. National infrastructures and **utilities** rely on computer-based systems and most electrical products include a computer and controlling software. Industrial manufacturing and distribution is completely computerized, as is the financial system. [1] Therefore, producing and maintaining software **cost-effectively** is essential for the functioning of national and international economies.



Software engineering is an engineering discipline whose focus is the cost-effective development of high-quality software systems. Software is abstract and intangible. It is not constrained by materials or governed by physical laws or by manufacturing processes. In some ways, this simplifies software engineering as there are no physical limitations on the potential of software. However, this lack of natural constraints means that software can easily become extremely complex and hence very difficult to understand.

The notion of software engineering was first proposed in 1968 at a conference held to discuss what was then called the “**software crisis**”. This software crisis **resulted** directly **from** the introduction of new computer hardware based on **integrated circuits**. Their power made **hitherto** unrealizable computer applications a feasible proposition. The resulting software was orders of magnitude larger and more complex than previous software systems.

Early experience in building these systems showed that informal software development was not good enough. Major projects were sometimes years late. The software cost much more than predicted, was unreliable, was difficult to maintain and performed poorly. Software development was in crisis. Hardware costs were **tumbling** whilst software costs were rising rapidly. New techniques and methods were needed to control the complexity **inherent** in large software systems.

These techniques have become part of software engineering and are now widely used. However, as our ability to produce software has increased, so has the complexity of the software systems

that we need. [2] New technologies resulting from the **convergence** of computers and communication systems and complex graphical user interfaces place new demands on software engineers. As many companies still do not apply software engineering techniques effectively, too many projects still produce software that is unreliable, delivered late and over budget.

We have made tremendous progress since 1968 and that the development of software engineering has **markedly** improved our software. We have a much better understanding of the activities involved in software development. We have developed effective methods of software specification, design and implementation. New **notations** and tools reduce the effort required to produce large and complex systems.

We know now that there is no single “ideal approach” to software engineering. The wide **diversity** of different types of systems and organizations that use these systems means that we need a diversity of approaches to software development. However, fundamental notions of process and system organization **underlie** all of these techniques, and these are the **essence** of software engineering.

Software engineers can be **rightly** proud of their achievements. Without complex software we would not have explored space, would not have the Internet and modern telecommunications, and all forms of travel would be more dangerous and expensive. Software engineering has contributed a great deal, and as the discipline matures, its contributions in the 21st century will be even greater.

Words

virtually

[ˈvɜ:tʃuəli] adv. 事实上, 实质上

utility

[ju:'tɪlɪti] n. 公用事业, 公共事业设备

cost-effective

[ˈkɒsti'fektɪv] adj. 有成本效益的, 划算的

notion

[ˈnəʊʃən] n. 概念, 观念, 看法

hitherto

[hɪðə'tu:] adv. 迄今, 至今

tumble

[ˈtʌmbəl] v. 使倒下, 搅乱

inherent

[ɪn'hɪərənt] adj. 固有的, 内在的, 与生俱来的

convergence

[kən'vɜ:dʒəns] n. 一体化, 集中, 收敛

markedly

[ˈmɑ:ktli] adv. 显著地, 明显地

notation

[nəʊ'teɪʃən] n. 符号

diversity

[daɪ'vɜ:sɪti] n. 差异, 多样性

underlie

[ʌndə'laɪ] v. 构成……的基础, 位于……之下

essence

[ˈesns] n. 本质, 实质

rightly

[ˈraɪtli] adv. 确实地

Phrases

software crisis

软件危机

result from

由……引起

integrated circuits

集成电路

Complex Sentences

[1] **Original:** Industrial manufacturing and distribution is completely computerized, as is the financial system.

Translation: 工业制造和配送完全用计算机处理, 财政系统也一样。

[2] **Original:** As our ability to produce software has increased, so has the complexity of the software systems that we need.

Translation: 当我们编制软件的能力增加的同时, 所需软件系统的复杂度也随之增加。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. The focus of software engineering is the rapid development of complex software systems.
- ___ 2. The notion of software engineering was first proposed in a paper in 1968.
- ___ 3. This software crisis resulted directly from the development of computer hardware.
- ___ 4. New notations and tools contribute to higher efficiency and less workload in producing large and complex software systems.
- ___ 5. As an engineering discipline, software engineering has matured adequately today.

II. Choose the best answer to each of the following questions.

1. Which of the following descriptions is not the characteristic of software?
 - (A) Abstract and intangible
 - (B) Not constrained by materials
 - (C) Not governed by physical laws or by manufacturing processes
 - (D) Easy to understand and simple to produce as there are no physical limitations
2. What problem(s) existed widely in informal software development in the early years?
 - (A) Over schedule
 - (B) Cost much more than budget
 - (C) Difficult to maintain
 - (D) All of the above
3. Which of the following statements is wrong about the techniques in software engineering?
 - (A) Techniques are needed to control the complexity of the large software systems.
 - (B) Techniques are the essence of software engineering.

- (C) Techniques are now widely used in software engineering.
- (D) New technologies bring new challenges to software engineers continually.

III. Translation

1. New technologies resulting from the convergence of computers and communication systems and complex graphical user interfaces place new demands on software engineers.
2. The wide diversity of different types of systems and organizations that use these systems means that we need a diversity of approaches to software development.

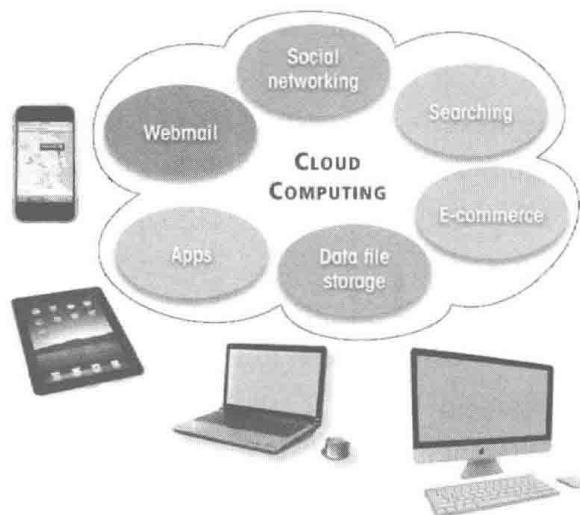
➤ Section B: Cloud Computing 云计算

There is a lot of buzz about **cloud computing** these days, and its scope continues to evolve. Cloud computing is a concept, rather than a specific technology. It is the idea that consumers use their computers or handheld devices to access applications, storage, and other computing resources supplied by Internet-based servers, rather than from their local devices.

In general, the term cloud computing refers to computing in which tasks are performed by a “cloud” of servers, typically via the Internet. This type of network has been used for several years to create the supercomputer-level power needed for research and other power-hungry applications, but it was more typically referred to as **grid computing** in this context. Today, cloud computing typically refers to accessing Web-based applications and data using a personal computer, mobile phone, or any other Internet-enabled device. The concept of cloud computing is that apps and data are available any time, from anywhere, and on any device. For example, you use cloud computing capabilities when you store or access documents, photos, videos, and other media online; use programs and apps online (i.e., E-mail, **productivity**, games, etc.); and share ideas, opinions, and content with others online (i.e., social networking sites).

While many of today’s cloud applications (such as Google Apps, Windows Live, Facebook, and YouTube) are consumer-oriented, other cloud applications (such as computing power or storage space available **on demand**, and online sales or service applications) are designed specifically for **businesses**. Businesses often use applications available in the **public cloud**; they also frequently create a **private cloud** just for data and applications belonging to their company.

The biggest advantages of cloud computing include the ability to access data from anywhere the user has access to an active Internet connection and, since data is stored online instead of on the device being used, the



data is safe if the device is lost, stolen, or damaged. In addition, Web-based applications are often less expensive than installed software. Disadvantages of cloud computing include a possible reduction in performance of applications if they run more slowly via the cloud than they would run if installed locally, and the potentially high expense related to data transfer for companies and individuals using high-bandwidth applications. [1] There are also security concerns about how safe the stored online data is from unauthorized access and data loss. Despite the potential risks, many believe that cloud computing is the **wave** of the future.

Home and business users choose cloud computing for a variety of reasons:

- **Accessibility:** Data and/or applications are available worldwide from any computer or device with an Internet connection.
- **Cost savings:** The expense of software and **high-end** hardware, such as fast processors and high-capacity memory and storage devices, **shifts** away from the user.
- **Space savings:** Floor space required for servers, storage devices, and other hardware shifts away from the user.
- **Scalability:** It provides the flexibility to increase or decrease computing requirements as needed.

Cloud computing allows companies to **outsource**, or **contract** to third-party providers, elements of their information technology **infrastructure**. They pay only for the computing power, storage, bandwidth, and access to applications that they actually use. As a result, companies need not make large investments in equipment, or the staff to support it. Consumers and organizations rely on cloud computing services to manage IT infrastructure (Infrastructure as a Service), provide applications (Software as a Service), access online data (Data as a Service), and create applications using Web-based development platforms (Platform as a Service).

Words

buzz

[ˈbʌz] n. 时髦的 (词语、想法或活动)

productivity

[ˌprɒdʌkˈtɪvɪti] n. 办公软件, 生产率, 生产力

business

[ˈbiznis] n. 企业, 商业, 交易, 生意, 事务, 业务, 职业, 行业

wave

[weɪv] n. 汹涌的行动 (或思想) 态势, 涌现的人 (或事物), 波浪

high-end

[haɪ end] adj. (尤指电子产品) 最高档的, 高端的, 最昂贵的

shift

[ʃɪft] v. 摆脱掉, 改变, 去掉, 换挡

scalability

[ˌskeɪləˈbɪlɪti] n. 可扩展性, 可伸缩性, 可量测性

outsource

[ˈaʊtsɔːrs] v. 外包 (工程)

contract

[kənˈtrækt] v. 承包, 订契约

infrastructure

[ˌɪnfreɪstrʌktʃə] n. 基础设施, 基础建设

Phrases

cloud computing	云计算
grid computing	网格计算
on demand	一经要求, 点播
public cloud	公有云
private cloud	私有云

Complex Sentences

Original: Disadvantages of cloud computing include a possible reduction in performance of applications if they run more slowly via the cloud than they would run if installed locally, and the potentially high expense related to data transfer for companies and individuals using high-bandwidth applications.

Translation: 云计算的缺陷主要体现在, 当应用程序在云端运行速度低于本地运行速度时, 云计算技术可能会降低应用程序的性能, 并可能会因公司与个人之间数据传递所需的高带宽造成开销上升。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Cloud computing is a specific technology.
- ___ 2. Cloud computing is the idea that consumers use their computers or handheld devices to access applications, storage and other computing resources from their local devices.
- ___ 3. Cloud computing typically refers to accessing Web-based applications and data using a personal computer, mobile phone, or any other Internet-enabled device.
- ___ 4. Web-based applications are often more expensive than installed software.
- ___ 5. Home and business users do not choose cloud computing due to its high expense.

II. Choose the best answer to each of the following questions.

- 1. Which of the following is wrong about the cloud computing?
 - (A) Cloud computing is a concept.
 - (B) Accessing Web-based applications and data is by using a personal computer, mobile phone, or any other Internet-enabled device.
 - (C) Cloud computing is a specific technology.
 - (D) The concept of cloud computing is that apps and data are available any time, from anywhere, and on any device.
- 2. Which of the following is wrong about choosing cloud computing for home and business users?
 - (A) Cost savings

- (B) Space savings
 - (C) Scalability
 - (D) Localization
3. Which of the following is right about buss today?
- (A) Data processing
 - (B) Personal computing
 - (C) Network computing
 - (D) Cloud computing

III. Translation

1. It is the idea that consumers use their computers or handheld devices to access applications, storage, and other computing resources supplied by Internet-based servers, rather than from their local devices.
2. The biggest advantages of cloud computing include the ability to access data from anywhere the user has access to an active Internet connection and, since data is stored online instead of on the device being used, the data is safe if the device is lost, stolen, or damaged.

Part 3 Simulated Writing: Memo

This guide will help you solve your memo-writing problems by discussing what a memo is, describing the parts of memos, and providing examples and explanations that will make your memos more effective.

Audience and Purpose

Memos have a twofold purpose: they bring attention to problems and they solve problems. They accomplish their goals by informing the reader about new information like policy changes, price increases, or by persuading the reader to take an action, such as attend a meeting, or change a current production procedure. Regardless of the specific goal, memos are most effective when they connect the purpose of the writer with the interests and needs of the reader.

Choose the audience of the memo wisely. Ensure that all of the people that the memo is addressed to need to read the memo. If it is an issue involving only one person, do not send the memo to the entire office. Also, be certain that material is not too sensitive to put in a memo; sometimes the best forms of communication are face-to-face interaction or a phone call. Memos are most effectively used when sent to a small to moderate amount of people to communicate company or job objectives.

Parts of a Memo

Standard memos are divided into segments to organize the information and to help achieve the

writer's purpose.

- **Heading**

The heading segment follows this general format:

TO: (readers' names and job titles)

FROM: (your name and job title)

DATE: (complete and current date)

SUBJECT: (what the memo is about, highlighted in some way)

Make sure you address the reader by his or her correct name and job title. And be specific and concise in your subject line.

- **Opening**

The purpose of a memo is usually found in the opening paragraph and includes: the purpose of the memo, the context and problem, and the specific assignment or task. Before indulging the reader with details and the context, give the reader a brief overview of what the memo will be about. Choosing how specific your introduction will depend on your memo plan style. The more direct the memo plan, the more explicit the introduction should be. Including the purpose of the memo will help clarify the reason the audience should read this document. The introduction should be brief, and should be approximately the length of a short paragraph.

- **Content**

The context is the event, circumstance, or background of the problem you are solving. You may use a paragraph or a few sentences to establish the background and state the problem. Include only what your reader needs, and be sure it is clear.

- **Task**

One essential portion of a memo is the task statement where you should describe what you are doing to help to solve the problem. Include only as much information as is needed by the decision-makers in the context, but be convincing that a real problem exists. Do not ramble on with insignificant details. If you are having trouble putting the task into words, consider whether you have clarified the situation. You may need to do more planning before you're ready to write your memo. Make sure your purpose-statement forecast divides your subject into the most important topics that the decision-maker needs.

- **Summary**

If your memo is longer than a page, you may want to include a separate summary segment. However, this section is not necessary for short memos and should not take up a significant amount of space. This segment provides a brief statement of the key recommendations you have

reached. These will help your reader understand the key points of the memo immediately. This segment may also include references to methods and sources you have used in your research.

- **Discussion**

The discussion segments are the longest portions of the memo, and are the parts in which you include all the details that support your ideas. Begin with the information that is most important. This may mean that you will start with key findings or recommendations. Start with your most general information and move to your specific or supporting facts. (Be sure to use the same format when including details: strongest to weakest.) The discussion segments include the supporting ideas, facts, and research that back up your argument in the memo. Include strong points and evidence to persuade the reader to follow your recommended actions. If this section is inadequate, the memo will not be as effective as it could be.

- **Closing**

After the reader has absorbed all of your information, you want to close with a courteous ending that states what action you want your reader to take. Make sure you consider how the reader will benefit from the desired actions and how you can make those actions easier.

- **Necessary Attachments**

Make sure you document your findings or provide detailed information whenever necessary. You can do this by attaching lists, graphs, tables, etc. at the end of your memo. Be sure to refer to your attachments in your memo and add a notation about what is attached below your closing.

Format

The format of a memo follows the general guidelines of business writing: A memo is usually a page or two long, should be single spaced and left justified. Instead of using indentations to show new paragraphs, skip a line between sentences. Business materials should be concise and easy to read.

You can help your reader understand your memo better by using headings for the summary and the discussion segments that follow it. Write headings that are short but that clarify the content of the segment. The major headings you choose are the ones that should be incorporated in your purpose-statement in the opening paragraph.

For easy reading, put important points or details into lists rather than paragraphs when possible. This will draw the readers' attention to the section and help the audience remember the information better. Using lists will help you be concise when writing a memo.

A Sample Memo

TO: All Lab Staff

FROM: John Smith, Lab Research Assistant

DATE: March 11, 2013

SUBJECT: New Carpet for Lab

This weekend, March 16-17, we will be re-carpeting the entire lab room. Therefore, before you leave on Friday, please make sure all wastebaskets, chairs, boxes and other items on the floor are moved into the hallway.

Also, clear your desktops and either your belongings in your desk drawers (which should be locked) or in boxes. Please label your boxes so that you can find your belongings easily in the next Monday morning.

Kind regards,

John Smith

Lab Research Assistant

Exercises

There are two scenarios as follows, please choose one and write a memo.

Scenario 1: In a team meeting, team members discussed the purchase of a coffee maker for the department, and you should report to your Department Manager, Mr. Wood, about your suggestion and the related research about this affair.

Scenario 2: In the weekly regular meeting, team members planned to hold a party for the New Year, and you should write a memo for this planning, including time, place, assignments, and so on.

Capturing the Requirements

需求获取

Part 1 Listening & Speaking

➤ Dialogue: Communication with Customers

(Kevin, Sharon and Jason come to the Four Seasons Hotel and have a meeting with Mr. White, the business manager of the hotel and the representative of the end user.)

Mr. White: Welcome to our hotel.

Kevin: Thank you for giving us the important information about your hotel management needs in your business requirements document. But I'm afraid that there are still some questions that we need to be clear about. In order that we can accomplish the system according to your meanings consistently, we would like to ask you a few questions.

Mr. White: Sure.

Sharon: Mr. White, I am drafting the specifications for the Four Seasons Hotel Management Information System, but I found that there may be an important function that is not clear.

Mr. White: Well, what's that?

Sharon: In the case that a customer **books** a room in your hotel, but doesn't **check in** on time, how do you **refund** his **deposit**, the whole, part or none? ^[1] It will relate to how to depict and **model** the **scenario** of the deposit refund and how to design the module.

Kevin: Yes. It is indeed significant for us to define a clear deposit refund mechanism in the system, because it will not only determine the design of the module itself, but also relate to other modules such as the total amount of the charge. After defining the work flow exactly, we will be able to know how to **deal with** it in the system accordingly.

[1] Replace with:
In the case that a customer books a dinner in your hotel, but doesn't check in on time, how do you deal with it?

Mr. White: Well, I see. It is indeed a necessary item in the room booking service. Now, let me explain to you our way in that situation. Usually, if a customer cancels his booking **up to** 24 hours before check-in, we will refund the deposit; and up to 12 hours before check-in, we will refund half, but if the cancellation comes within 12 hours, we don't refund anything.

Kevin: OK. Well, besides that, how is the rate of the refund deposit determined? Is there any difference between the VIP and common customers, or between the **high season** and **low season**?

Mr. White: In our hotel, there is no difference on the rate between the VIP and common customers, but we have **distinctive criteria** in different seasons. From May to October of every year, because of the high season, the timeline for booking cancellation is moved forward, 12 hours earlier than the low season which is from this November to next April. The customer must cancel his booking up to 36 hours before check-in if he determines and wants the whole deposit refunded. If the cancellation is between 36 and 24 hours prior to expected arrival, the refund would be half, and in case of within 24 hours, no refund.

Kevin: Yes, I've got it. And now, it seems clearer and more detailed. We will finish the requirements specification in the next three days and then send it to you this Friday by E-mail.

Mr. White: OK. No problem! Thanks.

Words

book

[buk] v. 登记, 预订

refund

[ri:'fʌnd] v. 退还, 偿还

deposit

[di'pɒzɪt] n. 押金, 预付金

model

['mɒdl] v. 建立模型

scenario

[si'nɑ:riəu] n. 场景, 某一特定情节

distinctive

[dis'tɪŋktɪv] adj. 区别性的, 特殊的

criterion

[kraɪ'tiəriən] n. 标准, 准则 (复数为 criteria)

Phrases

check in

登记, 报到

deal with

处理

up to

(时间上) 一直到

high season

旺季

low season

淡季

Abbreviations

VIP

Very Important Person

重要人物, 大人物

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ Listening Comprehension: Software Requirements

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which is the correct order of key steps in the software requirements stage according to this article?
 - Inception, elicitation, elaboration, negotiation, and validation
 - Elicitation, inception, elaboration, negotiation, and validation
 - Inception, elicitation, negotiation, elaboration, and validation
 - Elicitation, inception, negotiation, elaboration, and validation
- In which step are requirements further expanded into an analysis model?
 - Negotiation
 - Validation
 - Elaboration
 - Specification
- In which steps should software team work with other stakeholders in the software requirements stage?
 - Inception, negotiation, specification, and validation
 - Inception, elicitation, specification and validation
 - Inception, elicitation, negotiation, and validation
 - Inception, elaboration, negotiation, and specification

Words

conduct

[kən'dʌkt] v. 实施, 进行

distinct

[dis'tɪŋkt] adj. 不同的, 有区别的

inception

[in'sepʃən] n. 开端

elicitation

[i,lisi'teɪʃən] n. 导出, 启发

elaboration

[i,læbə'reɪʃən] n. 精化

negotiation

[ni,gəʊʃi'eɪʃən] n. 协商

validation

[væli'deɪʃən] n. 确认, 正确性检测

stakeholder

['steɪkhəʊldə] n. 利益相关者

Words

overriding

[əʊvə'raɪdɪŋ] adj. 首要的, 压倒一切的

class

[kla:s] n. 类

facilitate

[fə'siliti:teɪt] v. 使容易, 使便利, 促进

➤ **Dictation: The Difference between Customer and End-User**

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

Software ____1____ communicate with many different stakeholders, but customers and end-users have the most **significant impact** on the technical work that ____2____. In some ____3____ the customer and the end-user are one in the ____4____, but for many ____5____, the customer and the end-user are different people, ____6____ for different managers in different business ____7____.

A customer is the person or ____8____ who: (1) ____9____ requested the software to be ____10____; (2) defines ____11____ business objectives for the software; (3) provides basic product ____12____; and (4) **coordinates funding** for the project. In a ____13____ or system business, the customer is often the marketing ____14____. In an IT environment, the customer might be a business ____15____ or department.

On the ____16____, an end-user is the person or group who: (1) will ____17____ use the software that is built to ____18____ some business ____19____, and (2) will define ____20____ details of the software so that the business purposes can be achieved.

Words

significant

[sig'nifikənt] adj. 意义重大的, 重要的

coordinate

[kəu'ɔ:dɪnɪt] v. 调整, 整理

impact

['ɪmpækt] n. 影响, 效果

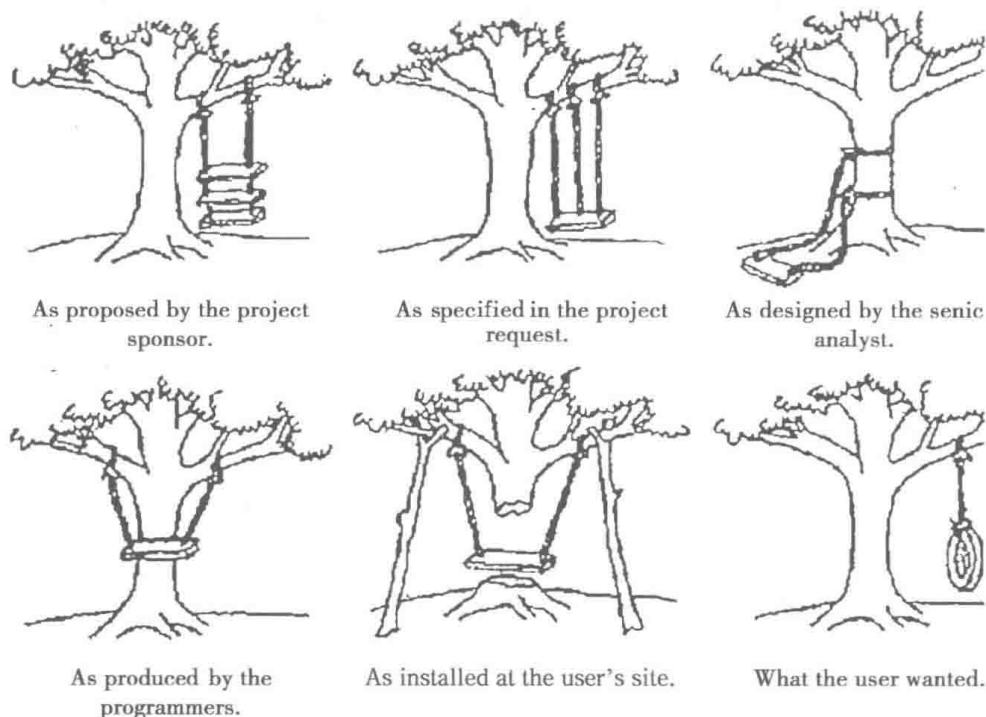
funding

['fʌndɪŋ] n. 提供资金

Part 2 Reading & Translation➤ **Section A: Software Requirements**

The main goal of the requirements phase is to produce the software requirements specification

(SRS), which accurately **captures** the client's requirements and which forms the basis of software development and validation. The basic reason for the difficulty in specifying software requirements comes from the fact that there are three **interested** parties — the client, the end users, and the software developer. The requirements document has to be such that the client and users can understand it easily and the developers can use it as a basis for software development. [1] Due to the diverse parties involved in software requirements specification, a communication gap exists. This makes the task of requirements specification difficult.



There are three basic activities in the requirements phase. The first is problem or requirement analysis. The goal of this activity is to understand such different aspects as the requirements of the problem, its **context**, and how it fits within the client's organization. The second activity is requirements specification, during which the understood problem is specified or written, producing the SRS. And the third activity is requirements validation, which is done to ensure that the requirements specified in the SRS are indeed what are **desired**.

There are three main approaches to analyze. Unstructured approaches rely on interaction between the analyst, customer, and user to **reveal** all the requirements (which are then **documented**). The second is the **modeling-oriented** approach, in which a model of the problem is built based on the available information. The model is useful in determining if the understanding is correct and in ensuring that all the requirements have been determined. Modeling may be **function-oriented** or **object-oriented**. The third approach is the prototyping approach in which a prototype is built to validate the correctness and completeness of requirements.

To satisfy its goals, a SRS should possess characteristics like completeness, **consistency**, unambiguous, **verifiable**, modifiable, etc. A good SRS should specify all the functions the

software needs to support, performance of the system, the **design constraints** that exist, and all the external interfaces.

One method for specifying the functional specifications that has become popular is the **use case** approach. With this approach the functionality of the system is specified through use cases, with each use case specifying the behavior of the system when a user interacts with it for achieving some goal. Each use case contains a normal scenario, as well as many exceptional scenarios, thereby providing the complete behavior of the system. Though use cases are meant for specification, as they are natural and story-like, by expressing them at different levels of abstraction they can also be used for problem analysis. [2]

For validation, the most commonly used method is reviewing or inspecting the requirements. In requirements inspections, the team of reviewers also includes a representative of the client to ensure that all requirements are captured.

Words

capture

['kæptʃə] v. 获得, 捕获

interested

['intrɪstɪd] adj. 有利害关系的, 有份儿的, 有关的

context

['kɒntekst] adj. 上下文, 环境, 背景

desire

[dɪ'zaɪə] v. 要求, 需要

reveal

[rɪ'vi:l] v. 展现, 显示, 暴露

document

['dɒkjumənt] v. 纪实性地描述

consistency

[kən'sɪstənsɪ] n. 一致性, 连贯性

verifiable

['verɪfaɪəbl] adj. 可证实的, 可检验的

Phrases

modeling-oriented

面向建模的

function-oriented

面向功能的

object-oriented

面向对象的

design constraint

设计约束

use case

用例

Abbreviations

SRS

Software Requirements Specification

软件需求规格说明

Complex Sentences

[1] Original: The requirements document has to be such that the client and users can understand it easily and the developers can use it as a basis for software development.

Translation: 需求文档必须能够让客户和用户容易理解, 并且能够让开发人员将其作为软件开发的基础来使用。

[2] Original: Though use cases are meant for specification, as they are natural and story-like, by expressing them at different levels of abstraction they can also be used for problem analysis.

Translation: 尽管用例是为规格说明而设定的, 但由于其自然性和故事性, 通过在不同的抽象级别上表达用例, 用例也可以用于问题分析。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Software requirements specification should accurately capture the client's requirements and form the basis of software development and validation.
- ___ 2. The goal of requirements validation is to understand such different aspects as the requirements of the problem, its context, and how it fits within the client's organization.
- ___ 3. The model and the prototype are both useful for the correctness and completeness of requirements.
- ___ 4. The use case approach has become one of the most popular methods for specifying the functional specifications.
- ___ 5. In requirements inspections, the representative of the client is responsible for ensuring that all requirements are captured.

II. Choose the best answer to each of the following questions.

- 1. Which statement is wrong about the SRS?
 - (A) It is the most important product in the requirements phase.
 - (B) The most direct reason for the difficulty in SRS is the diversity of the parties involved.
 - (C) A good SRS should specify not only all the functions of the software, but also other non-functional requirements.
 - (D) SRS should be inspected by the team of reviewers.
- 2. Which statement is right about the activity of requirement analysis?
 - (A) Prototype is a theoretic design to validate the correctness and completeness of requirements.
 - (B) During this activity, the understood problem is specified and written in SRS.
 - (C) There are three main different approaches for modeling.
 - (D) The main objective of this activity is to understand the problem.

3. Which statement is wrong about the use case approach?
- (A) Use cases specify the functionality of the system.
 - (B) Use cases can be used for different basic activities in the requirements phase.
 - (C) Each use case consists of many normal scenarios and many exceptional scenarios.
 - (D) Use case occurs when a user interacts with the system for achieving some goal.

III. Translation

The basic reason for the difficulty in specifying software requirements comes from the fact that there are three interested parties — the client, the end users, and the software developer.

➤ Section B: Mobile Web 移动互联网

The mobile Web refers to access to the World Wide Web, i.e. the use of browser-based Internet services, from a handheld mobile device, such as a smartphone or a **feature phone**, connected to a mobile network or other wireless network.

Traditionally, access to the Web has been via fixed-line services on large-screen laptops and desktop computers. However, the Web is becoming more accessible by portable and wireless devices. An early 2010 ITU (International Telecommunication Union) report said that with the current growth rates, Web access by people **on the go** — via laptops and smart mobile devices — is likely to exceed Web access from desktop computers within the next five years. The shift to mobile Web access has been **accelerating** with the rise since 2007 of larger multi-touch smartphones, and of multi-touch tablet computers since 2010. Both platforms provide better Internet access, screens, and mobile browsers - or application-based user Web experiences than previous generations of mobile devices have done. Web designers may work separately on such pages, or pages may be automatically converted as in Mobile Wikipedia.

The distinction between mobile Web applications(mobile apps) and **native applications** is anticipated to become increasingly blurred, as mobile browsers gain direct access to the hardware of mobile devices (including **accelerometers** and **GPS** chips), and the speed and abilities of browser-based applications improve. Persistent storage and access to sophisticated user interface graphics functions may further reduce the need for the development of platform-specific native applications.

Mobile Web access today still suffers from interoperability and usability problems. Interoperability issues stem from the **platform fragmentation** of mobile devices, mobile operating systems, and browsers. Usability problems are centered around the small physical size of the mobile phone **form factors** (limits on display resolution and user input/operating). Despite these shortcomings, many mobile developers choose to create apps using mobile Web. A June 2011 research on mobile development found mobile Web the third most used platform, **trailing** Android and iOS.

In an article in Communications of the ACM in April 2013, Web technologist Nicholas C. Zakas, **noted** that mobile phones in use in 2013 were more powerful than Apollo 11's 70 **lb** (32 kg) Apollo Guidance Computer used in the July 1969 lunar landing. However, in spite of their power, in 2013, mobile devices still suffer from Web performance with slow connections similar to the 1996 stage of Web development. Mobile devices with slower download request/response times, the latency of over-the-air data transmission, with “high-latency connections, slower CPUs, and less memory” force developers to rethink Web applications created for desktops with “wired connections, fast CPUs, and almost endless memory.” [1]



“Mobile Internet” refers to access to the Internet via a cellular telephone service provider. It is wireless access that can **hand off** to another radio tower while it is moving across the service area. It can refer an immobile device that stays connected to one tower, but this is not the meaning of “mobile” here. Wi-Fi and other better methods are commonly available for users not on the move. **Cellular base stations** are more expensive to provide than a wireless base station that connects directly to an Internet service provider, rather than through the telephone system.

A mobile phone, such as a smartphone, that connects to data or voice services without going through the cellular base station is not on mobile Internet. A laptop with a broadband modem and a cellular service provider subscription that is traveling on a bus through the city is on mobile Internet.

A mobile broadband modem “**tethers**” the smartphone to one or more computers or other end user devices to provide access to the Internet via the protocols that those cellular telephone service providers may offer.

Words

accelerating

[æk'selə'ret] v. (使)加快, (使)增速

Wikipedia

[wiki'pidi:ə] n. 维基百科(全书)

accelerometer

[æk'selə'ramitə] n. 加速计

interoperability

['intə,ɒpərə'bɪləti] n. 互用性, 协同工作的能力

Words**resolution**

[ˌrezəˈluʃən] n. 分辨率, 决心, 解决, 坚决

trail

[treil] v. (在比赛或其他竞赛中) 落后, 跟在……后面

note

[nəʊt] v. 指出, 对……加注释, 注意, 记录

lunar

[ˈluːnə] adj. 月球的

cellular

[ˈseljʊlə] adj. 蜂窝状的

tether

[ˈteðə] v. 束缚, 用绳拴住

Phrases

feature phone	功能手机
on the go	正在进行中, 忙碌中
native applications	本地应用
platform fragmentation	平台的碎片
form factor	(电子产品等的) 物理尺寸和形状, 规格
hand off	手递手传
base station	基站

Abbreviations

ITU	International Telecommunication Union	国际电信联盟
GPS	Global Positioning System	全球(卫星)定位系统
ACM	Association for Computing Machinery	美国计算机协会
lb	<拉> libra(e) (= pound(s))	磅

Complex Sentences

Original: Mobile devices with slower download request/response times, the latency of over-the-air data transmission, with “high-latency connections, slower CPUs, and less memory” force developers to rethink Web applications created for desktops with “wired connections, fast CPUs, and almost endless memory.”

Translation: 移动设备更低的下载速度/反应时间、数据传输延迟, 以及“高延迟连接、

更低速的 CPU 和更少的内存”，迫使开发者重新考虑原本为台式计算机设计的“有线连接、高速 CPU 和大量内存”的互联网应用。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Traditionally, access to the Web has been via a smartphone or a feature phone.
- ___ 2. Android is the third most used platform, trailing mobile Web and iOS.
- ___ 3. A laptop with a broadband modem and a cellular service provider subscription that is traveling on a bus through the city is on mobile Internet.
- ___ 4. Web apps don't require local storage space, so you don't have to worry about them accumulating on your hard disk.
- ___ 5. The difference between Web apps and mobile apps is that mobile apps run on a remote computer, whereas Web apps run from the handheld device, so they have to be downloaded and installed.

II. Choose the best answer to each of the following questions.

- 1. Which of the following is wrong about the mobile Web?
 - (A) The mobile Web refers to access to the World Wide Web, i.e. the use of browser-based Internet services, from a handheld mobile device, such as a smartphone or a feature phone, connected to a mobile network or other wireless network.
 - (B) Web access by people on the go is via laptops or smart mobile devices.
 - (C) Mobile Web access today still suffers from interoperability and usability problems.
 - (D) iOS is the third most used platform, trailing mobile Web and Android.
- 2. Which is the first most used platform?
 - (A) iOS
 - (B) Android
 - (C) Mobile Web
 - (D) Windows phone
- 3. Which statement is wrong about the differences between Web apps and mobile apps?
 - (A) Web apps run on a remote computer.
 - (B) Mobile apps run from the handheld device so that they have to be downloaded and installed.
 - (C) Games and entertainment seem to dominate Web apps.
 - (D) Games and entertainment seem to dominate mobile apps.

III. Translation

- 1. The mobile Web refers to access to the World Wide Web, i.e. the use of browser-based Internet services, from a handheld mobile device, such as a smartphone or a feature phone,

connected to a mobile network or other wireless network.

2. A mobile broadband modem “tethers” the smartphone to one or more computers or other end user devices to provide access to the Internet via the protocols that those cellular telephone service providers may offer.

Part 3 Simulated Writing: Software Requirements Specification

The requirements specification covers exactly the same ground as the requirements definition, but from the perspective of the developers. Where the requirements definition is written in terms of the customer’s vocabulary, referring to objects, states, events, and activities in the customer’s world, the requirements specification is written in terms of the system’s interface. We accomplish this by rewriting the requirements so that they refer only to those real-world objects (states, events, actions) that are sensed or actuated by the proposed system:

- (1) In documenting the system’s interface, we describe all inputs and outputs in detail, including the sources of inputs, the destinations of outputs, the value ranges and data formats of input and output data, protocols governing the order in which certain inputs and outputs must be exchanged, window formats and organization, and any timing constraints. Note that the user interface is rarely the only system interface; the system may interact with other software components (e.g. a database), special-purpose hardware, the Internet, and so on.
- (2) Next, we restate the required functionality in terms of the interfaces’ inputs and outputs. We may use a functional notation or data-flow diagrams to map inputs to outputs, or use logic to document functions’ pre-conditions and post-conditions. We may use state machines or events traces to illustrate exact sequences of operations or exact orderings of inputs and outputs. We may use an entity-relationship diagram to collect related activities and operations into classes. In the end, the specification should be complete, meaning that it should specify an output for any feasible sequence of inputs. Thus, we include validity checks on inputs and system responses to exceptional situations, such as violated pre-conditions.
- (3) Finally, we devise some fit criteria for each of the customer’s quality requirements, so that we can conclusively demonstrate whether our system meets these quality requirements.

The result is a description of what the developers are supposed to produce, written in sufficient detail to distinguish between acceptable and unacceptable solutions, but without saying how the proposed system should be designed or implemented.

Software Requirements Specification Template

Software Requirements Specification For <Project Name>

<Version>

<Author>

Prepared for

...

Instructor:

Date

Revision History

Date	Description	Author	Comments
<date>	<Version 1>	<Your Name>	<First Revision>

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Your Name>	Team Leader	
	...	Instructor, ...	

★ *Table of Contents*

REVISION HISTORY

DOCUMENT APPROVAL

1. Introduction

- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, Acronyms, and Abbreviations
- 1.4 References
- 1.5 Overview

2. General Description

- 2.1 Product Perspective
- 2.2 Product Functions
- 2.3 User Characteristics
- 2.4 General Constraints
- 2.5 Assumptions and Dependencies

3. Specific Requirements

- 3.1 External Interface Requirements
 - 3.1.1 User Interfaces
 - 3.1.2 Hardware Interfaces
 - 3.1.3 Software Interfaces
 - 3.1.4 Communications Interfaces
- 3.2 Functional Requirements
 - 3.2.1 <Functional Requirement or Feature #1>
 - 3.2.2 <Functional Requirement or Feature #2>
- 3.3 Use Cases
 - 3.3.1 Use Case #1
 - 3.3.2 Use Case #2
- 3.4 Classes / Objects
 - 3.4.1 <Class / Object #1>
 - 3.4.2 <Class / Object #2>
- 3.5 Non-Functional Requirements
 - 3.5.1 Performance
 - 3.5.2 Reliability
 - 3.5.3 Availability
 - 3.5.4 Security

3.5.5 Maintainability

3.5.6 Portability

3.6 Inverse Requirements

3.7 Design Constraints

3.8 Logical Database Requirements

3.9 Other Requirements

4. Analysis Models

4.1 Sequence Diagrams

4.2 Data Flow Diagrams (DFD)

4.3 State-Transition Diagrams (STD)

5. Change Management Process

A. Appendices

A.1 Appendix 1

A.2 Appendix 2

1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document (Note: the following subsection annotations are largely taken from the IEEE Guide to SRS).

1.1 Purpose

What is the purpose of this SRS and the (intended) audience for which it is written.

1.2 Scope

This subsection should:

- (1) Identify the software product(s) to be produced by name; for example, Host DBMS, Report Generator, etc.
- (2) Explain what the software product(s) will do, and, if necessary, will not do.
- (3) Describe the application of the software being specified. As a portion of this, it should:
 - (a) Describe all relevant benefits, objectives, and goals as precisely as possible. For example, to say that one goal is to provide effective reporting capabilities is not as good as saying parameter-driven, user-definable reports with a 2 h turnaround and on-line entry of user parameters.
 - (b) Be consistent with similar statements in higher-level specifications (for example, the System Requirement Specification), if they exist. What is the scope of this software product?

1.3 Definitions, Acronyms, and Abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

1.4 References

This subsection should:

- (1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.
- (2) Identify each document by title, report number-if applicable-date, and publishing organization.
- (3) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

1.5 Overview

This subsection should:

- (1) Describe what the rest of the SRS contains.
- (2) Explain how the SRS is organized.

2. General Description

This section of the SRS should describe the general factors that affect the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

2.1 Product Perspective

This subsection of the SRS puts the product into perspective with other related products or projects (See the IEEE Guide to SRS for more details).

2.2 Product Functions

This subsection of the SRS should provide a summary of the functions that the software will perform.

2.3 User Characteristics

This subsection of the SRS should describe those general characteristics of the eventual users of the product that will affect the specific requirements (See the IEEE Guide to SRS for more details).

2.4 General Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options for designing the system (See the IEEE Guide to SRS for a partial list of possible general constraints).

2.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will

be embodied within Section 2, but this section will give the requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- Correct
- Traceable (both forward and backward to prior/future artifacts)
- Unambiguous
- Verifiable (i.e., testable)
- Prioritized (with respect to importance and/or stability)
- Complete
- Consistent
- Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefully organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communications Interfaces

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

3.2.2 <Functional Requirement or Feature #2>

...

3.3 Use Cases

3.3.1 Use Case #1

3.3.2 Use Case #2

...

3.4 Classes / Objects

3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

3.4.2 <Class / Object #2>

...

3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

3.5.1 Performance

3.5.2 Reliability

3.5.3 Availability

3.5.4 Security

3.5.5 Maintainability

3.5.6 Portability

3.6 Inverse Requirements

State any useful inverse requirements.

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities,

data retention, data integrity, and so on?

3.9 Other Requirements

Catch all sections for any additional requirements.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable for the SRS's requirements.

4.1 Sequence Diagrams

4.2 Data Flow Diagrams (DFD)

4.3 State-Transition Diagrams (STD)

5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved?

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1

A.2 Appendix 2

PS: This template can be tailored according to the different projects.

A Sample Software Requirement Specification

Software Requirements Specification
For
Web Publishing System

Version 1.0

Victor Lu (Team Leader)

Jack Liang

Jason Wang

Remonna Zeng

April 10, 2013

<<Any comments inside double brackets such as these are not part of this SRS but are comments upon this SRS example to help the reader understand the point being made.

Refer to the SRS Template for details on the purpose and rules for each section of this document >>

Table of Contents

1. Introduction

- 1.1 Purpose
- 1.2 Scope of Project
- 1.3 Glossary
- 1.4 References
- 1.5 Overview of Document

2. Overall Description

- 2.1 System Environment
- 2.2 Functional Requirements Specification

2.2.1 Reader Use Case

Use case: Search Article

2.2.2 Author Use Case

Use case: Submit Article

2.2.3 Reviewer Use Case

Use case: Submit Review

2.2.4 Editor Use Cases

Use case: Update Author

Use case: Update Reviewer

Use case: Update Article

Use case: Receive Article

Use case: Assign Reviewer

Use case: Receive Review

Use case: Check Status

Use case: Send Response

Use case: Send Copyright

Use case: Remove Article

Use case: Publish Article

2.3 User Characteristics

2.4 Non-Functional Requirements

3. Requirements Specification

3.1 External Interface Requirements

3.2 Functional Requirements

3.2.1 Search Article

3.2.2 Communicate

- 3.2.3 Add Author
- 3.2.4 Add Reviewer
- 3.2.5 Update Person
- 3.2.6 Update Article Status
- 3.2.7 Enter Communication
- 3.2.8 Assign Reviewer
- 3.2.9 Check Status
- 3.2.10 Send Communication
- 3.2.11 Publish Article
- 3.2.12 Remove Article
- 3.3 Detailed Non-Functional Requirements
 - 3.3.1 Logical Structure of the Data
 - 3.3.2 Security

Index

List of Figures

- Figure 1 System Environment
- Figure 2 Article Submission Process
- Figure 3 Editor Use Cases
- Figure 4 Logical Structure of the Article Manager Data

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Web Publishing System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Regional Historical Society for its approval.

1.2 Scope of Project

This software system will be a Web Publishing System for a local editor of a regional historical society. This system will be designed to maximize the editor's productivity by providing tools to assist in automating the article review and publishing process, which would otherwise have to be performed manually. By maximizing the editor's work efficiency and production the system will meet the editor's needs while remaining easy to understand and use.

More specifically, this system is designed to allow an editor to manage and communicate with a group of reviewers and authors to publish articles to a public Website. The software will facilitate communication between authors, reviewers, and the editor via E-mail. Preformatted reply forms are used in every stage of the articles' progress through the system to provide a uniform review process; the location of these forms is configurable via the application's maintenance options. The system also contains a relational database containing a list of Authors, Reviewers, and Articles.

1.3 Glossary

Term	Definition
Active Article	The document that is tracked by the system; it is a narrative that is planned to be posted to the public website
Author	Person submitting an article to be reviewed. In case of multiple authors, this term refers to the <i>principal author</i> , with whom all communication is made
Database	Collection of all the information monitored by this system
Editor	Person who receives articles, sends articles for review, and makes final judgments for publications
Field	A cell within a form
Historical Society Database	The existing membership database (also HS database)
Member	A member of the Historical Society listed in the HS database
Reader	Anyone visiting the site to read articles

续表

Term	Definition
Review	A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement
Reviewer	A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made in the article
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
Stakeholder	Any person with an interest in the project who is not a developer
User	Reviewer or Author

1.4 References

IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, 1998.

1.5 Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2. Overall Description

2.1 System Environment

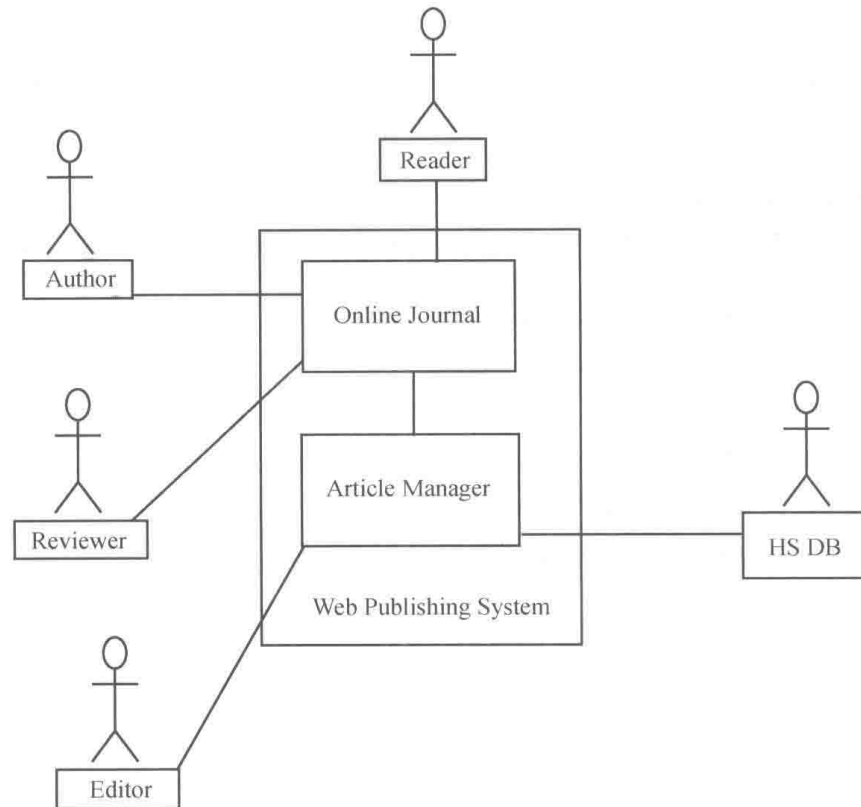


Figure 1 System Environment

The Web Publishing System has four active actors and one cooperating system (Figure 1).

The Author, Reader, or Reviewer accesses the Online Journal through the Internet. Any Author or Reviewer communication with the system is through E-mail. The Editor accesses the entire system directly. There is a link to the (existing) Historical Society.

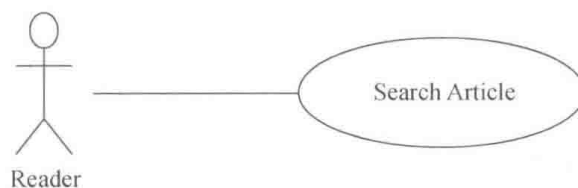
<< The division of the Web Publishing System into two component parts, the Online Journal and the Article Manager, is an example of using domain classes to make an explanation clearer. >>

2.2 Functional Requirements Specification

This section outlines the use cases for each of the active readers separately. The reader, the author and the reviewer have only one use case apiece while the editor is main actor in this system.

2.2.1 Reader Use Case

Use case: Search Article

Diagram:**Brief Description**

The Reader accesses the Online Journal Website, searches for an article and downloads it to his/her machine.

Initial Step-By-Step Description

Before this use case can be initiated, the Reader has already accessed the Online Journal Website.

1. The Reader chooses to search by author name, category, or keyword.
2. The system displays the choices to the Reader.
3. The Reader selects the article desired.
4. The system presents the abstract of the article to the Reader.
5. The Reader chooses to download the article.
6. The system provides the requested article.

Xref: Section 3.2.1, Search Article

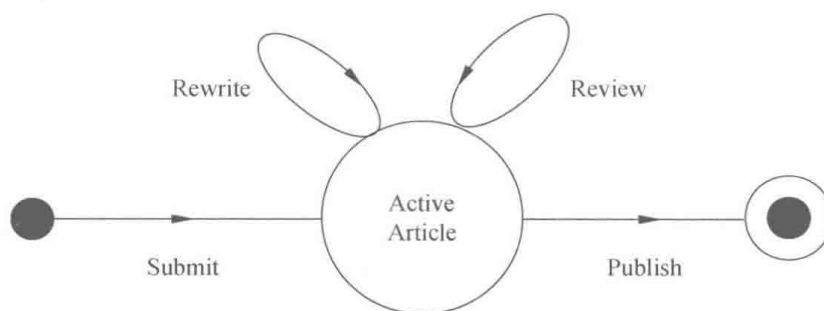


Figure 2 Article Submission Process

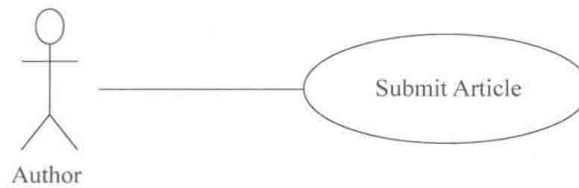
The *Article Submission Process* state-transition diagram (Figure 2) summarizes the use cases listed below. An Author submits an article for consideration. The Editor enters it into the system and assigns it to and sends it to at least three reviewers. The Reviewers return their comments, which are used by the Editor to make a decision on the article. Either the article is accepted as written, declined, or the Author is asked to make some changes based on the reviews. If it is accepted, possibly after a revision, the Editor sends a copyright form to the Author. When that form is returned, the article is published to the Online Journal. Not shown in the above is the removal of a declined article from the system.

2.2.2 Author Use Case

In case of multiple authors, this term refers to the *principal author*, with whom all communication is made.

Use case: Submit Article

Diagram:



Brief Description

The Author either submits an original article or resubmits an edited article.

Initial Step-By-Step Description

Before this use case can be initiated, the Author has already connected to the Online Journal Website.

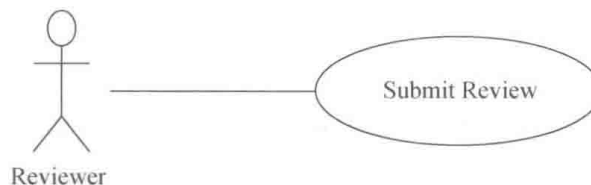
1. The Author chooses the *Email Editor* button.
2. The System uses the *sendto* HTML tag to bring up the user's E-mail system.
3. The Author fills in the Subject line and attaches the files as directed and E-mails them.
4. The System generates and sends an E-mail acknowledgement.

Xref: Section 3.2.2, Communicate

2.2.3 Reviewer Use Case

Use case: Submit Review

Diagram:



Brief Description

The Reviewer submits a review of an article.

Initial Step-By-Step Description

Before this use case can be initiated, the Reviewer has already connected to the Online Journal Website.

1. The Reviewer chooses the *Email Editor* button.
2. The System uses the *sendto* HTML tag to bring up the user's E-mail system.
3. The Reviewer fills in the Subject line and attaches the file as directed and E-mail it.
4. The System generates and sends an E-mail acknowledgement.

Xref: Section 3.2.2, Communicate

2.2.4 Editor Use Cases

The Editor has the following sets of use cases (Figure 3):

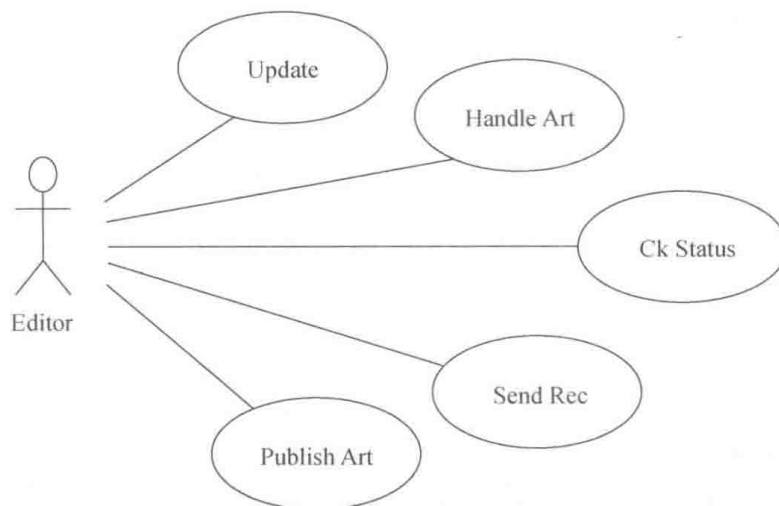
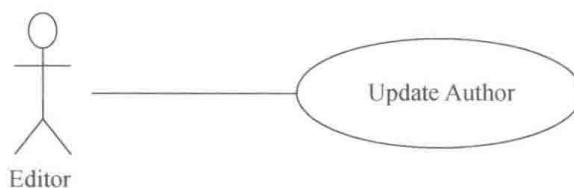


Figure 3 Editor Use Cases

Update Information use cases

Use case: Update Author

Diagram:



Brief Description

The Editor enters a new Author or updates information about a current Author.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

1. The Editor selects to *Add/Update Author*.
2. The system presents a choice of adding or updating.
3. The Editor chooses to add or to update.
4. If the Editor is updating an Author, the system presents a list of authors to choose from and presents a grid filling in with the information; else the system presents a blank grid.
5. The Editor fills in the information and submits the form.
6. The system verifies the information and returns the Editor to the Article Manager main page.

Xref: Section 3.2.3, Add Author; Section 3.2.5 Update Person

Use case: Update Reviewer

Diagram:**Brief Description**

The Editor enters a new Reviewer or updates information about a current Reviewer.

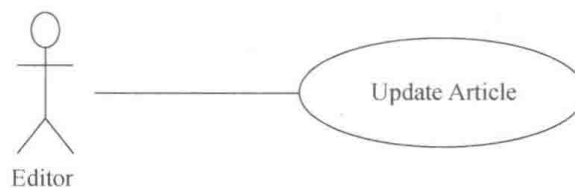
Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

1. The Editor selects to *Add/Update Reviewer*.
2. The system presents a choice of adding or updating.
3. The Editor chooses to add or to update.
4. The system links to the Historical Society Database.
5. If the Editor is updating a Reviewer, the system presents a grid with the information about the Reviewer; else the system presents list of members for the editor to select a Reviewer and presents a grid for the person selected.
6. The Editor fills in the information and submits the form.
7. The system verifies the information and returns the Editor to the Article Manager main page.

Xref: Section 3.2.4, Add Reviewer; Section 3.2.5, Update Person

Use case: Update Article

Diagram:**Brief Description**

The Editor enters information about an existing article.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

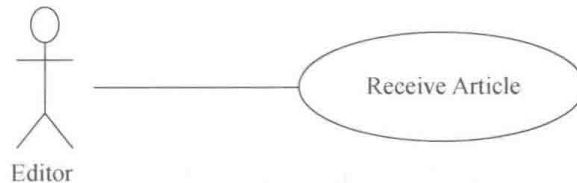
1. The Editor selects to *Update Article*.
2. The system presents a list of active articles.
3. The system presents the information about the chosen article.
4. The Editor updates and submits the form.
5. The system verifies the information and returns the Editor to the Article Manager main page.

Xref: Section 3.2.6, Update Article Status

Handle Article use cases

Use case: Receive Article

Diagram:



Brief Description

The Editor enters a new or revised article into the system.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager and has a file containing the article available.

1. The Editor selects to *Receive Article*.
2. The system presents a choice of entering a new article or updating an existing article.
3. The Editor chooses to add or to update.
4. If the Editor is updating an article, the system presents a list of articles to choose from and presents a grid for filling with the information; else the system presents a blank grid.
5. The Editor fills in the information and submits the form.
6. The system verifies the information and returns the Editor to the Article Manager main page.

Xref: Section 3.2.7, Enter Communication

Use case: Assign Reviewer

This use case extends the *Update Article* use case.

Diagram:



Brief Description

The Editor assigns one or more reviewers to an article.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

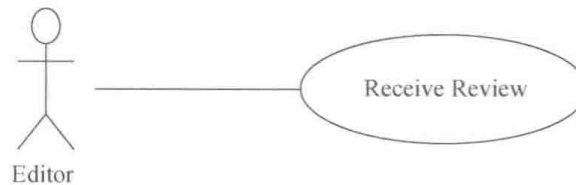
1. The Editor selects to *Assign Reviewer*.
2. The system presents a list of Reviewers with their status (see data description in section 3.3 below).
3. The Editor selects a Reviewer.
4. The system verifies that the person is still an active member using the Historical Society Database.
5. The Editor repeats steps 3 and 4 until sufficient reviewers are assigned.
6. The system E-mails the Reviewers, attaching the article and requesting that they do the review.
7. The system returns the Editor to the *Update Article* use case.

Xref: Section 3.2.8, Assign Reviewer

Use case: Receive Review

This use case extends the *Update Article* use case.

Diagram:



Brief Description

The Editor enters a review into the system.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

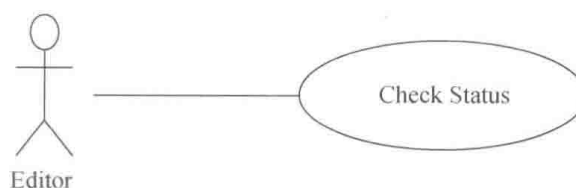
1. The Editor selects to *Receive Review*.
2. The system presents a grid for filling with the information.
3. The Editor fills in the information and submits the form.
4. The system verifies the information and returns the Editor to the Article Manager main page.

Xref: Section 3.2.7, Enter Communication

Check Status use case:

Use case: Check Status

Diagram:



Brief Description

The Editor checks the status of all active articles.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

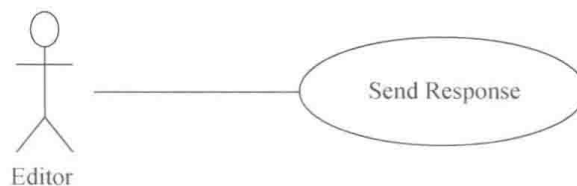
1. The Editor selects to *Check Status*.
2. The system returns a scrollable list of all active articles with their status (see data description in section 3.3 below).
3. The system returns the Editor to the Article Manager main page.

Xref: Section 3.2.9, Check Status

Send Recommendation use cases:

Use case: Send Response

This use case extends the *Update Article* use case.

Diagram:**Brief Description**

The Editor sends a response to an Author.

Initial Step-By-Step Description

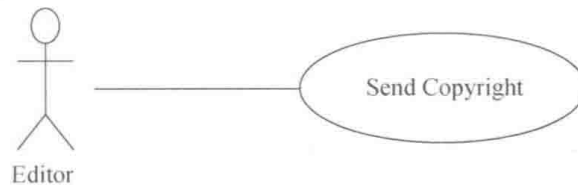
Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

1. The Editor selects to *Send Response*.
2. The system calls the E-mail system and puts the Author's E-mail address in the Recipient line and the name of the article on the subject line.
3. The Editor fills out the E-mail text and sends the message.
4. The system returns the Editor to the Article Manager main page.

Xref: Section 3.2.10, Send Communication

Use case: Send Copyright

This use case extends the *Update Article* use case.

Diagram:**Brief Description**

The Editor sends a copyright form to an Author.

Initial Step-By-Step Description

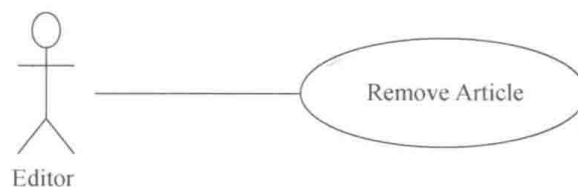
Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

1. The Editor selects to *Send Copyright*.
2. The system calls the E-mail system and puts the Author's E-mail address in the Recipient line, the name of the article on the subject line, and attaches the copyright form.
3. The Editor fills out the E-mail text and sends the message.
4. The system returns the Editor to the Article Manager main page.

Xref: Section 3.2.10, Send Communication

Use case: Remove Article

This use case extends the *Update Article* use case.

Diagram:**Brief Description**

The Editor removes an article from the active category.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

1. The Editor selects to remove an article from the active database.
2. The system provides a list of articles with the status of each.
3. The Editor selects an article for removal.
4. The system removes the article from the active article database and returns the Editor to the Article Manager main page.

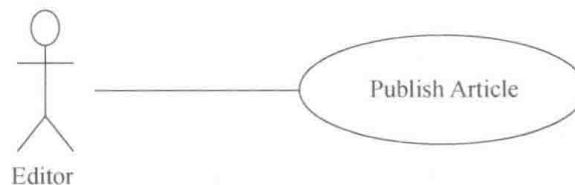
Xref: Section 3.2.12, Remove Article

Publish Article use case:

Use case: Publish Article

This use case extends the *Update Article* use case.

Diagram:



Brief Description

The Editor transfers an accepted article to the Online Journal.

Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the article using the *Update Article* use case.

1. The Editor selects to *Publish Article*.
2. The system transfers the article to the Online Journal and updates the search information there.
3. The system removes the article from the active article database and returns the Editor to the Article Manager home page.

Xref: Section 3.2.11, Publish Article

<< Since three of the actors only have one use case each, the summary diagram only involves the Editor. Adapt the rules to the needs of the document rather than adapt the document to fit the rules. >>

2.3 User Characteristics

The Reader is expected to be Internet literate and be able to use a search engine. The main screen of the Online Journal Website will have the search function and a link to “Author/Reviewer Information.”

The Author and Reviewer are expected to be Internet literate and to be able to use E-mail with attachments.

The Editor is expected to be Windows literate and to be able to use button, pull-down menus, and similar tools.

The detailed look of these pages is discussed in section 3.2 below.

2.4 Non-Functional Requirements

The Online Journal will be on a server with high speed Internet capability. The physical machine to be used will be determined by the Historical Society. The software developed here assumes the use

of a tool such as Tomcat for connection between the Web pages and the database. The speed of the Reader's connection will depend on the hardware used rather than characteristics of this system. The Article Manager will run on the editor's PC and will contain an Access database. Access is already installed on this computer and in a Windows operating system.

3. Requirements Specification

3.1 External Interface Requirements

The only link to an external system is the link to the Historical Society (HS) Database to verify the membership of a Reviewer. The Editor believes that a society member is much more likely to be an effective reviewer and has imposed a membership requirement for a Reviewer. The HS Database fields of interest to the Web Publishing Systems are member's name, membership (ID) number, and E-mail address (an optional field for the HS Database).

The *Assign Reviewer* use case sends the Reviewer ID to the HS Database and a Boolean is returned denoting membership status. The *Update Reviewer* use case requests a list of member names, membership numbers and (optional) E-mail addresses when adding a new Reviewer. It returns a Boolean for membership status when updating a Reviewer.

3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

3.2.1 Search Article

Use Case Name	Search Article
XRef	Section 2.2.1, Search Article SDD, Section 7.1
Trigger	The Reader assesses the Online Journal Website
Pre-condition	The Web is displayed with grids for searching
Basic Path	1. The Reader chooses how to search the Web site. The choices are by Author, by Category, and by Keyword. 2. If the search is by Author, the system creates and presents an alphabetical list of all authors in the database. In the case of an article with multiple authors, each is contained in the list. 3. The Reader selects an author. 4. The system creates and presents a list of all articles by that author in the database. 5. The Reader selects an article. 6. The system displays the Abstract for the article. 7. The Reader selects to download the article or to return to the article list or to the previous list

续表

Alternative Paths	In step 2, if the Reader selects to search by category, the system creates and presents a list of all categories in the database. 1. The Reader selects a category. 2. The system creates and presents a list of all articles in that category in the database. Return to step 5. In step 2, if the Reader selects to search by keyword, the system presents a dialog box to enter the keyword or phrase. 3. The Reader enters a keyword or phrase. 4. The system searches the Abstracts for all articles with that keyword or phrase and creates and presents a list of all such articles in the database. Return to step 5
Post-condition	The selected article is downloaded to the client machine
Exception Paths	The Reader may abandon the search at any time
Other	The categories list is generated from the information provided when article are published and not predefined in the Online Journal database

3.2.2 Communicate

Use Case Name	Communicate
XRef	Section 2.2.2, Submit Article; Section 2.2.3, Submit Review SDD, Section 7.2
Trigger	The user selects a <i>mailto</i> link
Pre-condition	The user is on the <i>Communicate</i> page linked from the Online Journal main page.
Basic Path	This use case uses the <i>mailto</i> HTML tag. This invokes the client E-mail facility
Alternative Paths	If the user prefers to use his or her own E-mail directly, sufficient information will be contained on the Web page to do so
Post-condition	The message is sent
Exception Paths	The attempt may be abandoned at any time
Other	None

3.2.3 Add Author

Use Case Name	Add Author
XRef	Section 2.2.4, Update Author SDD, Section 7.3
Trigger	The Editor selects to add a new author to the database
Pre-condition	The Editor has accessed the Article Manager main screen

续表

Basic Path	1. The system presents a blank grid to enter the author information. 2. The Editor enters the information and submits the form. 3. The system checks that the name and E-mail address fields are not blank and updates the database
Alternative Paths	If in step 2, either field is blank, the Editor is instructed to add an entry. No validation for correctness is made
Post-condition	The Author has been added to the database
Exception Paths	The Editor may abandon the operation at any time
Other	The author information includes the name, mailing address, and E-mail address

3.2.4 Add Reviewer

Use Case Name	Add Reviewer
XRef	Section 2.2.4, Update Reviewer SDD, Section 7.4
Trigger	The Editor selects to add a new reviewer to the database
Pre-condition	The Editor has accessed the Article Manager main screen.
Basic Path	1. The system accesses the Historical Society (HS) database and presents an alphabetical list of the society members. 2. The Editor selects a person. 3. The system transfers the member information from the HS database to the Article Manager (AM) database. If there is no E-mail address in the HS database, the Editor is prompted for an entry in that field. 4. The information is entered into the AM database
Alternative Paths	In step 3, if there is no entry for the E-mail address in the HS database or on this grid, the Editor will be re-prompted for an entry. No validation for correctness is made
Post-condition	The Reviewer has been added to the database
Exception Paths	The Editor may abandon the operation at any time
Other	The Reviewer information includes name, membership number, mailing address, categories of interest, and E-mail address

3.2.5 Update Person

Use Case Name	Update Person
XRef	Sec 2.2.4 Update Author; Sec 2.2.4 Update Reviewer SDD, Section 7.5
Trigger	The Editor selects to update an author or reviewer and the person is already in the database

续表

Pre-condition	The Editor has accessed the Article Manager main screen
Basic Path	1. The Editor selects Author or Reviewer. 2. The system creates and presents an alphabetical list of people in the category. 3. The Editor selects a person to update. 4. The system presents the database information in grid form for modification. 5. The Editor updates the information and submits the form. 6. The system checks that required fields are not blank
Alternative Paths	In step 5, if any required field is blank, the Editor is instructed to add an entry. No validation for correctness is made
Post-condition	The database has been updated
Exception Paths	If the person is not already in the database, the use case is abandoned. In addition, the Editor may abandon the operation at any time
Other	This use case is not used when one of the other use cases is more appropriate, such as to add an article or a reviewer for an article

3.2.6 Update Article Status

Use Case Name	Update Article Status
XRef	Section 2.2.4, Update Article SDD, Section 7.6
Trigger	The Editor selects to update the status of an article in the database
Pre-condition	The Editor has accessed the Article Manager main screen and the article is already in the database
Basic Path	1. The system creates and presents an alphabetical list of all active articles. 2. The Editor selects the article to update. 3. The system presents the information about the article in grid format. 4. The Editor updates the information and resubmits the form
Alternative Paths	In step 4, the use case <i>Enter Communication</i> may be invoked
Post-condition	The database has been updated
Exception Paths	If the article is not already in the database, the use case is abandoned. In addition, the Editor may abandon the operation at any time

续表

Other	This use case can be used to add categories for an article, to correct typographical errors, or to remove a reviewer who has missed a deadline for returning a review. It may also be used to allow access to the named use case to enter an updated article or a review for an article
--------------	---

3.2.7 Enter Communication

Use Case Name	Enter Communication
XRef	Section 2.2.4, Receive Article; Section 2.2.4, Receive Review SDD, Section 7.7
Trigger	The Editor selects to add a document to the system
Pre-condition	The Editor has accessed the Article Manager main screen and has the file of the item to be entered available
Basic Path	1. The Editor selects the article using the 3.2.6, <i>Update Article Status</i> use case. 2. The Editor attaches the file to the grid presented and updates the respective information about the article. 3. When the Editor updates the article status to indicate that a review is returned, the respective entry in the Reviewer table is updated
Alternative Paths	None
Post-condition	The article entry is updated in the database
Exception Paths	The Editor may abandon the operation at any time
Other	This use case extends 3.2.6, <i>Update Article Status</i>

3.2.8 Assign Reviewer

Use Case Name	Assign Reviewer
XRef	Section 2.2.4, Assign Reviewer SDD, Section 7.8
Trigger	The Editor selects to assign a reviewer to an article
Pre-condition	The Editor has accessed the Article Manager main screen and the article is already in the database
Basic Path	1. The Editor selects the article using the 3.2.6, <i>Update Article Status</i> use case. 2. The system presents an alphabetical list of reviewers with their information. 3. The Editor selects a reviewer for the article. 4. The system updates the article database entry and E-mail the reviewer with the standard message and attaches the text of the article without author information. 5. The Editor has the option of repeating this use case from step 2

续表

Alternative Paths	None
Post-condition	At least one reviewer has been added to the article information and the appropriate communication has been sent
Exception Paths	The Editor may abandon the operation at any time
Other	This use case extends 3.2.6, <i>Update Article Status</i> . The Editor, prior to implementation of this use case, will provide the message text

3.2.9 Check Status

Use Case Name	Check Status
XRef	Section 2.2.4, Check Status SDD, Section 7.9
Trigger	The Editor has selected to check status of all active articles
Pre-condition	The Editor has accessed the Article Manager main screen
Basic Path	1. The system creates and presents a list of all active articles organized by their status. 2. The Editor may request to see the full information about an article
Alternative Paths	None
Post-condition	The requested information has been displayed
Exception Paths	The Editor may abandon the operation at any time
Other	The Editor may provide an enhanced list of status later. At present, the following categories must be provided: 1. Received but no further action taken. 2. Reviewers have been assigned but not all reviews are returned (include dates that reviewers were assigned and order by this criterion). 3. Reviews returned but no further action taken. 4. Recommendations for revision sent to Author but no response as of yet. 5. Author has revised article but no action has been taken. 6. Article has been accepted and copyright form has been sent. 7. Copyright form has been returned but article is not yet published. A published article is automatically removed from the active article list

3.2.10 Send Communication

Use Case Name	Send Communication
XRef	Section 2.2.4, Send Response; Section 2.2.4, Send Copyright SDD, Section 7.10
Trigger	The Editor selects to send a communication to an author

续表

Pre-condition	The Editor has accessed the Article Manager main screen
Basic Path	1. The system presents an alphabetical list of authors. 2. The Editor selects an author. 3. The system invokes the Editor's E-mail system entering the author's E-mail address into the <i>To:</i> entry. 4. The Editor uses the email facility
Alternative Paths	None
Post-condition	The communication has been sent
Exception Paths	The Editor may abandon the operation at any time
Other	The standard copyright form will be available in the Editor's directory for attaching to the E-mail message, if desired

3.2.11 Publish Article

Use Case Name	Publish Article
XRef	Section 2.2.4, Publish Article SDD, Section 7.11
Trigger	The Editor selects to transfer an approved article to the Online Journal
Pre-condition	The Editor has accessed the Article Manager main screen
Basic Path	1. The system creates and presents an alphabetical list of the active articles that are flagged as having their copyright form returned. 2. The Editor selects an article to publish. 3. The system accesses the Online Database and transfers the article and its accompanying information to the Online Journal database. 4. The article is removed from the active article database
Alternative Paths	None
Post-condition	The article is properly transferred
Exception Paths	The Editor may abandon the operation at any time
Other	Find out from the Editor to see if the article information should be archived somewhere

3.2.12 Remove Article

Use Case Name	Remove Article
XRef	Section 2.2.4, Remove Article SDD, Section 7.12

续表

Trigger	The Editor selects to remove an article from the active article database
Pre-condition	The Editor has accessed the Article Manager main screen
Basic Path	1. The system provides an alphabetized list of all active articles. 2. The Editor selects an article. 3. The system displays the information about the article and requires that the Editor confirm the deletion. 4. The Editor confirms the deletion
Alternative Paths	None
Post-condition	The article is removed from the database
Exception Paths	The Editor may abandon the operation at any time
Other	Find out from the Editor to see if the article and its information should be archived somewhere

3.3 Detailed Non-Functional Requirements

3.3.1 Logical Structure of the Data

The logical structure of the data to be stored in the internal Article Manager database is given below (Figure 4).

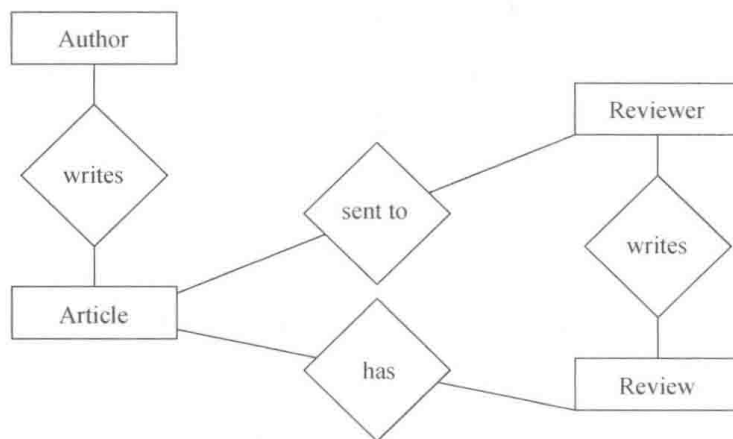


Figure 4 Logical Structure of the Article Manager Data

The data descriptions of each of these data entities are as follows:

Author Data Entity

Data Item	Type	Description	Comment
Name	Text	Name of principle author	
Email Address	Text	Internet address	
Article	Pointer	Article entity	May be several

Reviewer Data Entity

Data Item	Type	Description	Comment
Name	Text	Name of principle author	
ID	Integer	ID number of Historical Society member	Used as key in Historical Society Database
E-mail Address	Text	Internet address	
Article	Pointer	Article entity	May be several
Num Review	Integer	Review entity	Number of not returned reviews
History	Text	Comments on past performance	
Specialty	Category	Area of expertise	May be several

Review Data Entity

Data Item	Type	Description	Comment
Article	Pointer	Article entity	
Reviewer	Pointer	Reviewer entity	Single reviewer
Date Sent	Date	Date sent to reviewer	
Returned	Date	Date returned; null if not returned	
Contents	Text	Text of review	

Article Data Entity

Data Item	Type	Description	Comment
Name	Text	Name of Article	
Author	Pointer	Author entity	Name of principle author
Other Authors	Text	Other authors is any; else null	Not a pointer to an Author entity
Reviewer	Pointer	Reviewer entity	Will be several
Review	Pointer	Review entity	Set up when reviewer is set up
Contents	Text	Body of article	Contains Abstract as first paragraph.
Category	Text	Area of content	May be several
Accepted	Boolean	Article has been accepted for publication	Needs Copyright form returned
Copyright	Boolean	Copyright form has been returned	Not relevant unless Accepted is True.
Published	Boolean	Sent to Online Journal	Not relevant unless Accepted is True. Article is no longer active and does not appear in status checks.

The logical structure of the data to be stored in the Online Journal database on the server is as follows:

Published Article Entity

Data Item	Type	Description	Comment
Name	Text	Name of Article	
Author	Text	Name of one Author	May be several
Abstract	Text	Abstract of Article	Used for keyword search
Content	Text	Body of Article	
Category	Text	Area of content	May be several

3.3.2 Security

The server on which the Online Journal resides will have its own security to prevent unauthorized *write/delete* access. There is no restriction on *read* access. The use of E-mail by an Author or Reviewer is on the client systems and thus is external to the system.

The PC on which the Article Manager resides will have its own security. Only the Editor will have physical access to the machine and the program on it. There is no special protection built into this system other than to provide the Editor with *write* access to the Online Journal to publish an article.

Exercises

1. How many actors are there in this Web Publishing System?
2. What does the use case Editor do?
3. How many data items are there in the data entity Articles?

Planning the Project

项目计划

Part 1 Listening & Speaking

➤ Dialogue: Software Project Planning

Jason: So much work to do.

Kevin: I think we need a formal project plan as our guideline.

Sharon: Yes, the first one is the time, which is one of the most important factors for a project. We need a schedule, especially, the deadline of our project.

Kevin: We have 40 days in total; the requirements acquirement has taken us 5 days already, so there are 35 days left.

Jason: Oh, it sounds really urgent.

Sharon: It seems that we should begin to program as soon as possible, right?

Jason: Although coding is a very central part of a software project, the most important thing is that we must first establish a proper schedule to control our progress and assure our **deployment** on time, I think.

Kevin: Yes. In the requirements phase, we will spend another 3 days to depict, analyze and model the requirements. After that, we will spend 3 days to complete the **architectural design** and it will take 5 days to accomplish the **detailed design**. Because of the effort applied to software design, code should follow with relatively little difficulty, and can be done within one week I think. Testing and subsequent **debugging** can **account for** about 10 days of software development effort.

Sharon: Maybe we can draw our schedule using a **Gantt Chart**. It is a visual and effective tool for a project plan.

Kevin: Good idea.

Jason: Actually, testing should not be seen as an activity which starts only after

the coding phase is complete, with the limited purposes of defecting failures. ^[1] Indeed, planning for testing should start with the early stages of the requirement process, and test plans and procedures must be systematically and continuously developed, and possibly refined, as development proceeds. During coding, we can perform the unit testing at the same time. It will save much time and obtain better testing effects I think. Finally, we can perform a validation test by working with the customer to find out if the software developed is valid for the customer and make sure that the customer is getting what they asked for.

Sharon: By the way, we need three computers and must install the software which the customer requires with a uniform version as developing tools. That is Microsoft Visual Studio 2012 for a development platform and Microsoft SQL server 2012 as a database management system.

Jason: We need a network as well.

Kevin: Ok, I will prepare the development environment for us as soon as possible. Then, we must assign some management responsibilities to everyone. Sharon, you are responsible for document management ^[2]. Jason, you take charge of change management. And I will be in charge of the Software Quality Assurance. Ok?

Jason & Sharon: No problem.

Kevin: Ok, I have noted everything which we referred to just now, and will complete a project plan within two days.

[1] Replace with:

Software design is an **indispensable** process which has a significant impact on the next processes in the entire development.

[2] Replace with:

1. database management
2. configuration management
3. risk management

Words

deployment

[di'plɔɪmənt] n. 部署

debug

[di:'bʌg] v. 调试

indispensable

[,ɪndɪ'spensəbl] adj. 不可缺少的, 必需的

Phrases

architectural design

体系结构设计, 概要设计

detailed design

详细设计

account for

(在数量、比例方面) 占

Gantt Chart

甘特图

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

► Listening Comprehension: Software Project Planning

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- How many kinds of planning philosophies are mentioned in the article?
(A) Two
(B) Three
(C) Four
(D) Five
- How many questions are stated in Boehm's principle for leading project planning?
(A) Three
(B) Five
(C) Six
(D) Seven
- Which point of view is the most accordant with the idea of this article on the project planning?
(A) Planning every activities in the project as detailed as possible for their foreseeable ability
(B) Carrying out as early as possible regardless of planning, because even the best planning can be obviated by uncontrolled change as the work proceeds
(C) Adjusting different levels of details for activities according to their different locations in the project timeline
(D) Making a perfect plan which can evade changes that may come about during the work

Words

minimalist

['miniməlist] n. 最低限要求者

argue

['ɑ:gju:] v. 主张, 认为

obviate

['ɒbviət] v. 排除, 避免

traditionalist

[trə'diʃənəlist] n. 传统主义者, 墨守成规者

agilist

['ædʒaɪlist] n. 敏捷主义者, 机敏者

fruitless

['fru:tɪs] adj. 徒劳的, 无用的

recipe

['resipi] n. 诀窍

chaos

['keɪɔs] n. 混乱

moderation

['mɒdə'reɪʃən] n. 适度, 节制

paper

['peɪpə] n. 论文, 文章

state

[steɪt] v. 陈述, 说明

milestone

['maɪlstəʊn] n. 里程碑

Phrases

road map

线路图, (一步一步的) 详尽计划

in moderation

适中地

Abbreviations

W5HH

Why, What, When, Who, Where, How, How much

➤ Dictation: Four Variables in Projects

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

A project is a carefully defined set of ____ 1 ____ that use resources to achieve ____ 2 ____ goals and objectives. It is a ____ 3 ____ **endeavor** having specific start and completion dates as well as a managed ____ 4 ____ having a range of ____ 5 ____, budget and organizational constraints.

It is usually considered that there are four ____ 6 ____ variables we will ____ 7 ____ in software projects—cost, time, scope, and quality. These four variables affect a project together by ____ 8 ____ with each other: increased scope typically means more time and ____ 9 ____, a tight time **stress** could mean stronger finance support and **suffered** ____ 10 ____, too little money couldn't solve the customer's business problem ____ 11 ____ the scope, and a higher quality might deliver longer developing ____ 12 ____ and more cost.

It seems that there is not a simple relationship between them. For example, you can't just get software faster by spending more money. As the ____ 13 ____ goes, "Nine women cannot make a ____ 14 ____ in one month."

In his book **Extreme Programming** Explained: Embrace Change, Kent Beck says that the solution is to make the four variables ____ 15 ____ . If everyone — programmers, customers, and managers — can see all four variables, they can **consciously** ____ 16 ____ which variables to control. If they don't like the result ____ 17 ____ for fourth variable, they can change the ____ 18 ____, or they can ____ 19 ____ a different three variables to control for ____ 20 ____ ultimately the project objectives.

Words

endeavor

[in'devə] n. 努力, 尽力

suffer

['sʌfə] v. 受损失, 受害

stress

[stres] n. 压力, 紧迫

embrace

[im'breis] v. 掌握, 接受, 拥抱

Words**consciously**

['kɒnʃəsli] adv. 有意识地

ultimately

['ʌltimətli] adv. 最后, 终于, 根本

Phrases

Extreme Programming

极限编程

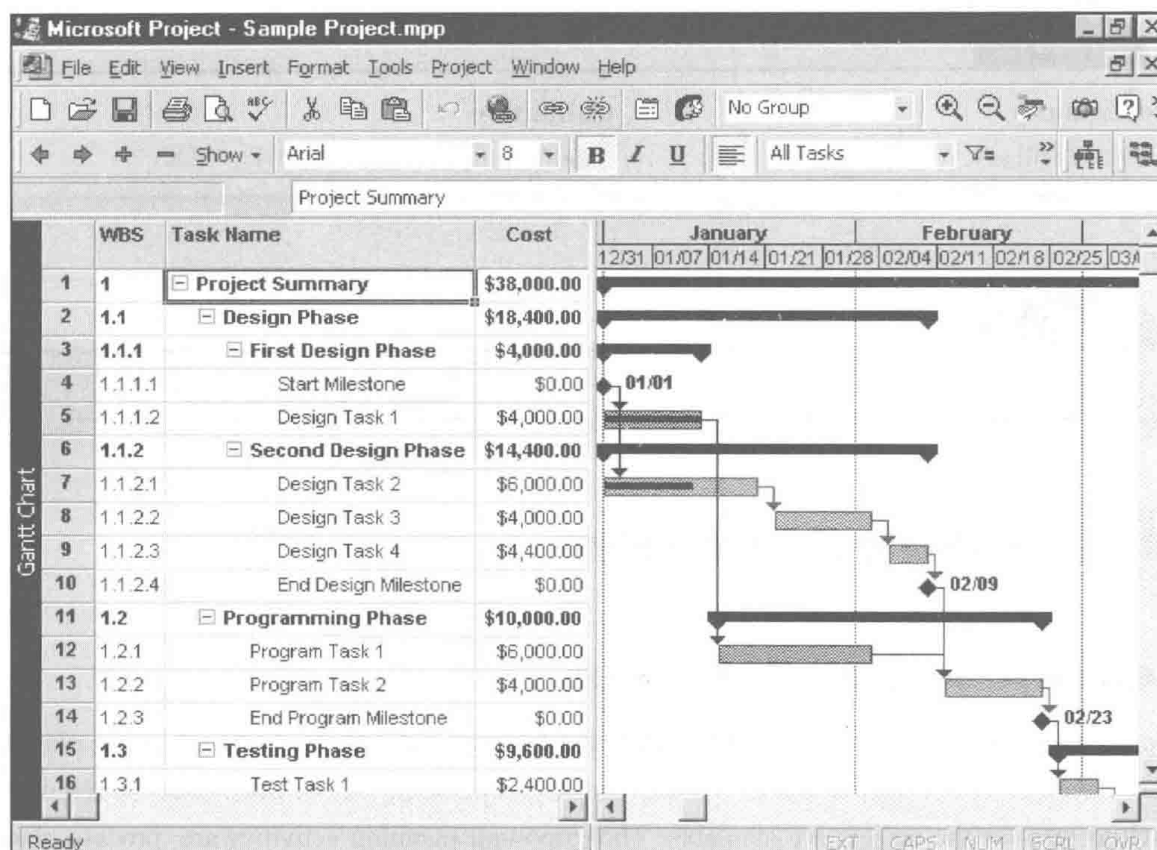
Part 2 Reading & Translating**➤ Section A: Software Project Plan**

A proper project plan is an important **ingredient** for a successful project. Without proper planning, a software development project is unlikely to succeed. Good planning can be done after the requirements and architecture for the project are available. The important planning activities are: process planning, effort estimation, scheduling and **staffing** planning, quality planning, **configuration** management planning, project monitoring planning, and risk management.

Process planning generally involves selecting a proper process model and **tailoring** it to suit the project needs. In effort estimation, overall effort requirement for the project and the **breakup** of the effort for different phases is estimated. In a **top-down approach**, total effort is first estimated, frequently from the estimate of the size, and then effort for different phases or tasks is determined. In a **bottom-up approach**, the main tasks in the project are identified, and effort for them is estimated first. From the effort estimates of the tasks, the overall estimate is obtained.

The overall schedule and the major milestones of a project depend on the effort estimate and the staffing level in the project and simple models can be used to get a rough estimate of schedule from effort. Often, an overall schedule is determined using a model, and then adjusted to meet the project needs and constraints. The detailed schedule is one in which the tasks are broken into smaller, schedulable tasks, and then assigned to specific team members, while preserving the overall schedule and effort estimates. [1] The detailed schedule is the most live document of project planning as it lists the tasks that have to be done; any changes in the project plan must be reflected suitably in the detailed schedule.

Quality plans are important for ensuring that the final product is of high quality. The project quality plan identifies all the **V&V** activities that have to be performed at different stages in the development, and how they are to be performed.



The goal of configuration management is to control the changes that take place during the project. The configuration management plan identifies the **configuration items** which will be controlled, and specifies the procedures to accomplish this and how access is to be controlled.

Risks are those events which may or may not occur, but if they do occur, they have a negative impact on the project. To meet project goals even under the presence of risks requires proper risk management. Risk management requires that risks be identified, analyzed, and **prioritized**. Then risk **mitigation** plans are made and performed to minimize the effect of the highest priority risks.

For a plan to be successfully implemented it is essential that the project be monitored carefully. Activity level monitoring, status reports, and milestone analysis are the **mechanisms** that are often used. For analysis and reports, the actual effort, schedule, **defects**, and size should be measured. With these measurements, it is possible to monitor the performance of a project **with respect to** its plan. And based on this monitoring, **actions** can be **taken** to correct the course of execution, if the need **arises**.

Overall, project planning **lays out** the path the project should follow in order to achieve the project objectives. It specifies all the tasks that the project members should perform, and specifies who will do what, in how much time, and when in order to execute this plan. With a detailed plan, what remains to be done is to execute the plan, which is done through the rest of the project. Of course, plans never remain unchanged, as things do not always work as planned.

With proper monitoring in place, these situations can be identified and plans changed accordingly. Basic project planning principles and techniques can be used for plan modification also.

Words

ingredient

[in'gri:diənt] n. 成分, 因素

staff

['stɑ:f] v. 配置职员

configuration

[kən'figju'reiʃən] n. 配置

tailor

['teilə] v. 剪裁, 适应

suit

[sju:t] v. 适合, 合乎……的要求

breakup

['breikʌp] n. 分解

live

[laiv] adj. 生动的

prioritize

[prai'ɔ:ritaiz] v. 把……区分优先次序

mitigation

[,miti'geiʃən] n. 缓解, 减轻

mechanism

['mekənizəm] n. 机制

defect

[di'fekt] n. 缺陷

course

[kɔ:s] n. 过程, 进程

arise

[ə'raiz] v. 出现, 产生

Phrases

top-down approach

自顶向下方法

bottom-up approach

自底向上方法

configuration items

配置项

with respect to

就……而论, 关于

take action

采取行动, 着手

lay out

划定(路线), 布置, 安排

Abbreviations

V&V

Verification and Validation

验证和确认

Complex Sentences

Original: The detailed schedule is one in which the tasks are broken into smaller, schedulable tasks, and then assigned to specific team members, while preserving the overall schedule and effort estimates.

Translation: 详细进度是指, 在保持总体进度和工作量估算的条件下, 将任务分解为更小、可安排的任务, 然后将其分派给特定的团队成员。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. There are seven important activities in the project planning.
- ___ 2. Generally, there are two main steps involved in process planning, that is selecting a proper process model and tailoring the project to suit the model chosen.
- ___ 3. In software engineering, the term “activity” is in the level with more details than the term “task” is.
- ___ 4. Both the effort estimate and the staffing level in a project are the bases of the overall schedule and the major milestones of the project.
- ___ 5. Status reports is one of the mechanisms that are often used in project monitoring.

II. Choose the best answer to each of the following questions.

1. Which statement is right about the project planning?
 - (A) Planning is an important ingredient for a successful project, so it must be done at the very beginning, before any other activities in the project.
 - (B) Project planning arranges the path the project should follow in order to achieve the project objectives.
 - (C) Establishing a good project plan is the most essential for implementing the plan successfully.
 - (D) Because of the potential risks brought by changes, a good project plan always avoids changes, so that things always work as planned.
2. What does the word “live” means in the context of “The detailed schedule is the most live document of project planning” in the third paragraph?
 - (A) Involving team members who are physically present, because the tasks must be assigned to specific team members in the detailed schedule.
 - (B) Guidable, because the detailed schedule lists the tasks that have to be done, as the guidance of the daily work.
 - (C) Operative, because the detailed schedule must consider any changes in the project plan suitably.
 - (D) Not specific, because the detailed schedule does not have to be made due to the potential frequent changes.
3. Which statement is wrong about the following different activities in the project plan?
 - (A) Models that can be used in process planning and schedule planning are different.
 - (B) Configuration management is used to control the changes that occur during the project.
 - (C) Risk mitigation plans are made and performed as the subsequence of the risk management plan.
 - (D) Measurement is a most important mechanism for monitoring the performance of a project with respect to its plan.

III. Translation

1. To meet project goals even under the presence of risks requires proper risk management.
2. With proper monitoring in place, these situations can be identified and plans changed accordingly.

➤ Section B: Big Data 大数据

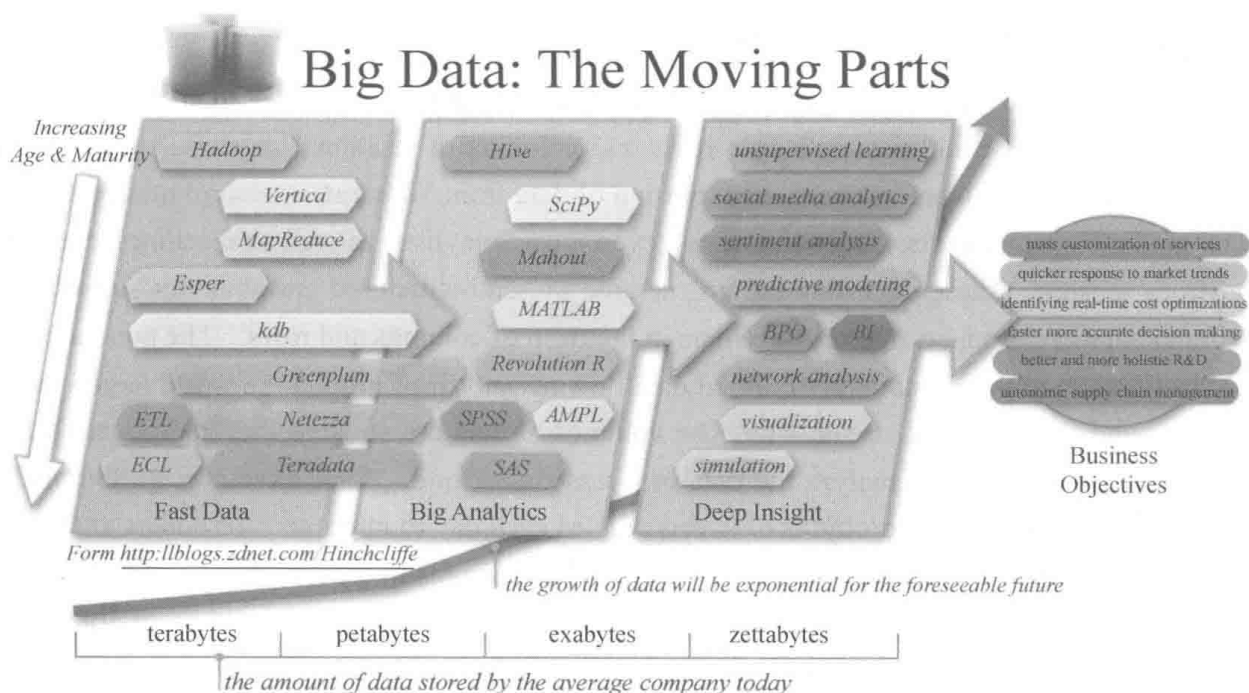
Up until about five years ago, most data collected by organizations consisted of structured transaction data that could easily fit into rows and columns of relational database management systems. Since then, there has been an explosion of data from Web traffic, E-mail messages, and social media content(**tweets**, status messages), even music playlists, as well as machine-generated data from sensors. These data may be unstructured or semi-structured and thus not suitable for relational database products that organize data in the form of columns and rows. The popular term “big data” refers to this **avalanche** of digital data flowing into firms around the world largely from Web sites and Internet click stream data. The volumes of data are so large that traditional **DBMS** cannot capture, store, and analyze the data in a reasonable time. Some examples of “big data” challenges are analyzing 12 **terabytes** of tweets created each day to improve your understanding of consumer **sentiment** towards your products; 100 million E-mails in order to place appropriate ads alongside the E-mail messages; or 500 million call detail records to find patterns of fraud and **churn**. [1] Big data and the tools needed to deal with it really started with Google and other search engines. Google’s problem: it has to deal with 500 million searches a day, and within milliseconds, display search results and place ads. For fun, do a search on “big data” and you’ll see Google respond with more than 1 billion results in 38 milliseconds (about a third of a second). That’s much faster than you can read this sentence! Big data usually refers to data in the **petabyte** and **exabyte** range — in other words, billions to trillions of records, all from different sources. Big data are produced in much larger quantities and much more rapidly than traditional data. Even though “tweets” are limited to 140 characters each, Twitter generates more than 8 terabytes of data daily. According to the IDC technology research firm, data is more than doubling every two years, so the amount of data available to organizations is **skyrocketing**. **Making sense** out of it quickly in order to gain a market advantage is critical.

Businesses are interested in big data because they contain more patterns and interesting **anomalies** than smaller data sets, with the potential to provide new insights into customer behavior, weather patterns, financial market activity, or other phenomena. [2]

However, to derive business value from these data, organizations need new technologies and tools capable of managing and analyzing nontraditional data along with their traditional enterprise data.

To handle unstructured and semi-structured data in vast quantities, as well as structured data, organizations are using Hadoop. Hadoop is an open source software **framework** managed by the

Apache Software Foundation that enables distributed parallel processing of huge amounts of data across inexpensive computers. It breaks a big data problem down into sub-problems, distributes them among up to thousands of inexpensive computer processing nodes, and then combines the result into a smaller data set that is easier to analyze. You've probably used Hadoop to find the best **airfare** on the Internet, get directions to a restaurant, search on Google, or connect with a friend on Facebook.



Hadoop can process large quantities of any kind of data, including structured transactional data, loosely structured data such as Facebook and Twitter feeds, complex data such as Web server log files, and unstructured audio and video data. Hadoop runs on a **cluster** of inexpensive servers, and processors can be added or removed as needed. Companies use Hadoop to analyze very large volumes of data as well as for a **staging** area for unstructured and semi-structured data before they are loaded into a **data warehouse**. Facebook stores much of its data on its massive Hadoop cluster, which holds an estimated 100 petabytes about 10,000 times more information than the Library of Congress. Yahoo uses Hadoop to track user behavior so it can modify its home page to fit user interests. Life sciences research firm NextBio uses Hadoop and HBase to process data for pharmaceutical companies conducting **genomic** research. Top database vendors such as IBM, Hewlett-Packard, Oracle, and Microsoft have their own Hadoop software distributions. Other vendors offer tools for moving data into and out of Hadoop or for analyzing data within Hadoop.

Words

tweet

[ˈtwi:t] n. 微博, 消息

avalanche

[ˈævələ:ntʃ] n. 纷至沓来, 蜂拥而至, 雪崩

terabyte

[ˈterəbaɪt] n. 太 (10¹²) 字节

sentiment

[ˈsentɪmənt] n. 意见, 观点, 情绪

Words**churn**

[tʃɜ:n] n. 粗制滥造, 搅拌

petabyte['petəbait] n. 拍 (10^{15}) 字节**exabyte**['eksə, bait] n. 艾 (10^{24}) 字节**skyrocket**

['skai'rɒkit] v. 突升, 猛涨

anomaly

[ə'nɒməli] n. 异常, 反常

framework

['freimwɜ:k] n. 构架, 框架

airfare

['eəfeə] n. 飞机票价

cluster

['klʌstə] n. 群, 簇

stage

[steɪdʒ] v. 展现, 举行

genomic

[dʒi:nəʊmɪk] adj. 染色体组的

Phrases

make sense

理解, 有意义, 是明智的

data warehouse

数据仓库

Abbreviations

DBMS

DataBase Management System

数据库管理系统

Complex Sentences

[1] **Original:** Some examples of “big data” challenges are analyzing 12 terabytes of tweets created each day to improve your understanding of consumer sentiment towards your products; 100 million E-mails in order to place appropriate ads alongside the E-mail messages; or 500 million call detail records to find patterns of fraud and churn

Translation: 大数据带来的挑战如: 通过分析每天产生的 12 太字节 (10^{12} B) 的微博数据来更好地了解用户对产品的看法; 分析 1 亿封电子邮件以便在邮件信息旁边投放合适的广告; 或分析 5 亿条详细的呼叫记录来寻找诈骗和粗制滥造的方式。

[2] **Original:** Businesses are interested in big data because they contain more patterns and interesting anomalies than smaller data sets, with the potential to provide new insights into customer behavior, weather patterns, financial market activity, or other phenomena.

Translation: 企业对大数据感兴趣, 是因为它们与更小的数据集相比包含了更多的模式与有趣的特例, 因而具备提供新的针对顾客行为、天气模式、金融市场活动或其他现象相关信息的潜力。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Traditional data are produced in much larger quantities and much more rapidly than big data.
- ___ 2. Hadoop can handle unstructured and semi-structured data in vast quantities, as well as structured data.
- ___ 3. Companies use Excel to analyze very large volumes of data as well as for a staging area for unstructured and semi-structured data before they are loaded into a data warehouse.
- ___ 4. Businesses are interested in big data because they contain a lot of useful information.
- ___ 5. Big data refers to huge collections of data that are easy to process, analyze, and manage using conventional database tools.

II. Choose the best answer to each of the following questions.

- 1. Which tool can be used to process the big data?
 - (A) Word
 - (B) Powerpoint
 - (C) Excell
 - (D) Hadoop
- 2. Which company was a pioneer to start big data and the tools needed to deal with it?
 - (A) Lenovo
 - (B) Google
 - (C) Sharp
 - (D) Dell
- 3. Which tool would a bank manager use to quickly find out how many fix-rate and how many variable-rate mortgage loans were approved by the bank during each of the past four years?
 - (A) A data mine
 - (B) A data warehouse
 - (C) An OLAP executive dashboard
 - (D) Big data

III. Translation

However, to derive business value from these data, organizations need new technologies and tools capable of managing and analyzing nontraditional data along with their traditional enterprise data.

Part 3 Simulated Writing: Software Project Plan

To communicate risk analysis and management, project cost estimates, schedule, and organization

to our customers, we usually write a document called a project plan. The plan puts in writing the customer's needs, as well as what we hope to do to meet them. The customer can refer to the plan for information about activities in the development process, making it easy to follow the project's progress during development. We can also use the plan to confirm with the customer any assumptions we are making, especially about cost and schedule.

A good project plan includes the following items:

1. project scope
2. project schedule
3. project team organization
4. technical description of the proposed system
5. project standards, procedures, and proposed techniques and tools
6. quality assurance plan
7. configuration management plan
8. documentation plan
9. data management plan
10. resource management plan
11. test plan
12. training plan
13. security plan
14. risk management plan
15. maintenance plan

The scope defines the system boundary, explaining what will be included in the system and what will not be included. It assures the customer that we understand what is wanted. The schedule can be expressed using a work breakdown structure, the deliverables, and a timeline to show what will be happening at each point during the project life cycle. A Gantt Chart can be useful in illustrating the parallel nature of some of the development tasks.

The project plan also lists the people on the development team, how they are organized, and what they will be doing. As we have seen, not everyone is needed all the time during the project, so the plan usually contains a resource allocation chart to show staffing levels at different times.

Writing a technical description forces us to answer questions and address issues as we anticipate how development will proceed. This description lists hardware and software, including compilers, interfaces, and special-purpose equipment or software. Any special restrictions on cabling, execution time, response time, security, or other aspects of functionality or performance are documented in the plan.

Software Project Plan Template

Software Project Plan

for

<Name of Project>

<Version>

<author>

<date>

Table of Contents

Build the table of contents here. Insert it when you finish your document.

1. Introduction

This section of the Software Project Plan(SPP) provides an overview of the project.

1.1 Project Overview

Include a concise summary of the project objectives, major work activities, major milestones, required resources, and budget. Describe the relationship of this project to other projects, if appropriate. Provide a reference to the official statement of product requirements.

1.2 Project Deliverables

List the primary deliverables for the customer, the delivery dates, delivery locations, and quantities required satisfying the terms of the project agreement.

1.3 Evolution of the SPP

Describe how this plan will be completed, disseminated, and put under change control. Describe how both scheduled and unscheduled updates will be handled.

1.4 Reference Materials

Provide a complete list of all documents and other sources of information referenced in the plan. Include for each the title, report number, date, author, and publishing organization.

1.5 Definitions and Acronyms

Define or provide references to the definition of all terms and acronyms required to properly interpret the SPP.

2. Project Organization

This section specifies the process model for the project and its organizational structure.

2.1 Process Model

Specify the life cycle model to be used for this project or refer to an organizational standard model that will be followed. The process model must include roles, activities, entry criteria and exit criteria for project initiation, product development, product release, and project termination.

2.2 Organizational Structure

Describe the internal management structure of the project, as well as how the project relates to the rest of the organization. It is recommended that charts be used to show the lines of authority.

2.3 Organizational Interfaces

Describe the administrative and managerial interfaces between the project and the primary entities with which it interacts. A table may be a useful way to represent this(Figure 1).

Organization	Liaison	Contact Information
Customer: <name>	<name>	<phone, E-mail, etc.>
Subcontractor: <name>		
Software Quality Assurance		
Software Configuration Management		
<etc.>		

Figure 1 Project Interfaces

2.4 Project Responsibilities

Identify and state the nature of each major project function and activity, and identify the individuals who are responsible for those functions and activities. Tables of functions and activities may be used to depict project responsibilities(Figure 2).

Role	Description	Person
Project Manager	leads project team; responsible for project deliverables	<name>
Technical Team Leader(s)	<define as locally used>	<name>
<etc.>	<etc.>	

Figure 2 Project Responsibilities

3. Managerial Process

This section of the SPP specifies the management process for this project.

3.1 Management Objectives and Priorities

Describe the philosophy, goals, and priorities for managing this project. A flexibility matrix might be helpful in communicating what dimensions of the project are fixed, constrained and flexible. Each degree of flexibility column can contain only one "X"(Figure 3).

Project Dimension	Fixed	Constrained	Flexible
Cost		X	
Schedule	X		
Scope (functionality)			X

Figure 3 Flexibility Matrix

3.2 Assumptions, Dependencies, and Constraints

State the assumptions on which the project is based, any external events the project is dependent upon, and the constraints under which the project is to be conducted. Include an explicit statement of the relative priorities among meeting functionality, schedule, and budget for this project.

3.3 Risk Management

Describe the process to be used to identify, analyze, and manage the risk factors associated with the project. Describe mechanisms for tracking the various risk factors and implementing contingency plans. Risk factors that should be considered include contractual risks, technological risks, risks due to size and complexity of the product, risks in personnel acquisition and retention, and risks in achieving customer acceptance of the product. The specific risks for this project and the methods for managing them may be documented here or in another document included as an appendix or by reference.

3.4 Monitoring and Controlling Mechanisms

Define the reporting mechanisms, report formats, review and audit mechanisms, and other tools and techniques to be used in monitoring and controlling adherence to the SPP. Project monitoring should occur at the level of work packages. Include monitoring and controlling mechanisms for the project support functions (quality assurance, configuration management, documentation and training).

A table may be used to show the reporting and communication plan for the project. The communication table can show the regular reports and communication expected of the project, such as weekly status reports, regular reviews, or as-needed communication. The exact types of communication vary between groups, but it is useful to identify the planned means at the start of the project(Figure 4).

Information Communicated	From	To	Time Period
Status report	Project Team	Project Manager	Weekly
Status report	Project Manger	Software Manager, Project Team	Weekly
Project Review	Project Team	Software Manager	Monthly
<etc>			

Figure 4 Communication and Reporting Plan

3.5 Staffing Approach

Describe the types of skills required for the project, how appropriate personnel will be recruited, and any training required for project team members.

4. Technical Process

This section specifies the technical methods, tools, and techniques to be used on the project. It also includes identification of the work products and reviews to be held and the plans for the support group activities in user documentation, training, software quality assurance, and configuration management.

4.1 Methods, Tools, and Techniques

Identify the computing system(s), development method(s), standards, policies, procedures, team structure(s), programming language(s), and other notations, tools, techniques, and methods to be used to specify, design, build, test, integrate, document, deliver, modify or maintain the project deliverables.

4.2 Software Documentation

Specify the work products to be built for this project and the types of peer reviews to be held for those products. It may be useful to include a table that is adapted from the organization's standard collection of work products and reviews. Identify any relevant style guide, naming conventions and documentation formats. In either this documentation plan or the project schedule provide a summary of the schedule and resource requirements for the documentation effort.

To ensure that the implementation of the software satisfies the requirements, the following documentation is required as a minimum:

4.2.1 Software Requirements Specification (SRS)

The SRS clearly and precisely describes each of the essential requirements (functions, performances, design constraints, and attributes) of the software and the external interfaces. Each requirement is defined such that its achievement is capable of being objectively verified and validated by a prescribed method, for example, inspection, analysis, demonstration, or test.

4.2.2 Software Design Description (SDD)

The SDD describes the major components of the software design including databases and internal interfaces.

4.2.3 Software Test Plan

The Software Test Plan describes the methods to be used for testing at all levels of development and integration: requirements as expressed in the SRS, designs as expressed in the SDD, code as expressed in the implemented product. The test plan also describes the test procedures, test cases, and test results that are created during testing activities.

4.3 User Documentation

Describe how the user documentation will be planned and developed (This may be just a

reference to a plan being built by someone else). Include work planned for online as well as paper documentation, online help, network accessible files and support facilities.

4.4 Project Support Functions

Provide either directly or by reference, plans for the supporting functions for the software project. These functions may include, but are not limited to, configuration management, software quality assurance, and verification and validation. Plans for project support functions are developed to a level of detail consistent with the other sections of the SPP. In particular, the responsibilities, resource requirements, schedules and budgets for each supporting function must be specified. The nature and type of support functions required will vary from project to project, however, the absence of a software quality assurance, configuration management, or, verification and validation plan must be explicitly justified in project plans that do not include them.

5. Work Packages, Schedule, and Budget

Specify the work packages, dependency relationships, resource requirements, allocation of budget and resources to work packages, and a project schedule. Much of the content may be in appendices that are living documents, updated as the work proceeds.

5.1 Work Packages

Specify the work packages for the activities and tasks that must be completed in order to satisfy the project agreement. Each work package is uniquely identified. A diagram depicting the breakdown of project activities and tasks (a work breakdown structure) may be used to depict hierarchical relationships among work packages.

5.2 Dependencies

Specify the ordering relations among work packages to account for interdependencies among them and dependencies on external events. Techniques such as dependency lists, activity networks, and the critical path method may be used to depict dependencies among work packages.

5.3 Resource Requirements

Provide, as a function of time, estimates of the total resources required to complete the project. Numbers and types of personnel, computer time, support software, computer hardware, office and laboratory facilities, travel, and maintenance requirements for the project resources are typical resources that should be specified.

5.4 Budget and Resource Allocation

Specify the allocation of budget and resources to the various project functions, activities, and tasks.

5.5 Schedule

Provide the schedule for the various project functions, activities, and tasks, taking into account the precedence relations and the required milestone dates. Schedules may be expressed in absolute calendar time or in increments relative to a key project milestone.

6. Additional Components

Certain additional components may be required and may be appended as additional sections or subsections to the SPP. Additional items of importance on any particular project may include subcontractor management plans, security plans, independent verification and validation plans, training plans, hardware procurement plans, facilities plans, installation plans, data conversion plans, system transition plans, or the product maintenance plan.

6.1 Index

An index to the key terms and acronyms used throughout the SPP is optional, but recommended to improve usability of the SPP.

6.2 Appendices

Appendices may be included, either directly or by reference, to provide supporting details that could detract from the SPP if included in the body of the SPP. Suggested appendices include:

- A. Current Top 10 Risk Chart
- B. Current Project Work Breakdown Structure
- C. Current Detailed Project Schedule

PS: This template can be tailored according to the different projects.

A Sample Software Project Plan

Software Project Plan For **Web Publishing System** Version 1.0

Victor Lu(Team Leader)
Jack Liang
Jason Wang
Remonna Zeng
April 5, 2013

Table of Contents

1. Introduction

- 1.1 Scope of project
- 1.2 Major software functions
- 1.3 Major constraints
- 1.4 Purpose of document
- 1.5 Project deliverables
- 1.6 Glossary
- 1.7 References

2. Project Organization

- 2.1 Project schedule
- 2.2 Deliverables and milestones
- 2.3 Project estimation chart
- 2.4 People(project group)/ Staff organization
- 2.5 Management reporting and communication

3. Tracking and Control Mechanisms

- 3.1 Quality assurance and control
- 3.2 Change management and control

4. Method and Tool

5. Risk Management

1. Introduction

1.1 Scope of project

This software system will be a Web Publishing System for a local editor of a regional historical society. It will be designed to maximize the editor's productivity by providing tools to assist in automating the article review and publishing process, otherwise it has to be performed manually. By maximizing the editor's work efficiency and production, the system will meet the editor's needs while remaining easy to understand and use.

1.2 Major software functions

Eventually, editor could use this product to manage and communicate with a group of reviewers and authors to publish articles more efficiently. Moreover, the software could facilitate communication between authors, reviewers, and the editor via E-mails.

1.3 Major constraints

- Budget: \$50,000
- Time: two months
- Staff: 4 members in this group.

1.4 Purpose of document

This document describes the plan of the Web Publishing System project. It also provides cost, effort and time estimate of this project which is used to monitor and control the project to guide the project execution. The aim of these processes is to make sure to complete the project within budget as well as by the due date.

1.5 Project deliverables

All of the items listed in this subsection are the deliverables requested by the Web Publishing System.

Software documentation:

- Installation documentation
- End-user documentation

Project documentation:

- Software Requirements Specification(SRS)
- Software Design Specification(SDS)
- Software Project Plan(SPP)
- Software Test Plan (STP)
- Software Quality Assurance Plan(SQAP)

1.6 Glossary

Term	Definition
Author	Person submitting an article to be reviewed. In case of multiple authors, this term refers to the principal author, with whom all communication is made.
Database	Collection of all the information monitored by this system.
Editor	Person who receives articles, sends articles for review, and makes final judgments for publications.
Historical Society Database	The existing membership database (also HS database).
Member	A member of the Historical Society listed in the HS database.
Reader	Anyone visiting the site to read articles.
Review	A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement.
Reviewer	A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made in the article.

1.7 References

A00-00-02-00-2010.SPP

2. Project Organization

2.1 Project schedule

This section presents an overview of project tasks and schedule. The project will use the waterfall development approach for fulfilling the tasks of different phase on time.

2.2 Deliverables and milestones

Stage Of Development	Stage Completion Date	Milestone	Deliverable Completion Date
Planning	04/05/13	Quality Assurance Plan completed Project Plan completed	04/05/13 04/05/13
Requirements Definition	04/10/13	Draft Requirements Specification completed Draft Design Specification completed Requirements Specification (final) completed	04/08/13 04/09/13 04/10/13
Design (Functional & System)	04/22/13	Draft Training Plan completed Program and Database Specifications completed Design Specification (final) completed	04/15/13 04/18/13 04/22/13
Programming	05/10/13	Software (frontend and backend) completed System Test Plan completed	05/10/13 05/08/13
Integration & Testing	05/21/13	System Test Plan (final) completed Test Reports completed User's Guide completed	05/14/13 05/17/13 05/21/13
Installation & Acceptance	05/31/13	Maintenance Plan completed Project Closeout	05/23/13 05/31/13

2.3 Project estimation chart

Task Name	Duration	Staff
Allocate project resources	2 days	1
Establish project environment	2 days	2
Create software project plan	2 day	1
Define and develop software requirements	2 days	1
Create Software Requirements Specification(SRS)	1 day	1
Perform architectural design	3 days	2

续表

Task Name	Duration	Staff
Design the database	2 days	1
Design the interfaces	3 days	1
Create Software Design Specification(SDS)	1 day	3
Programming	14 days	3
Plan testing	2 days	2
Execute the tests	5 days	2
User manual development	1 day	2
Reapply another software lifecycle	8 days	4

The Project estimation chart. Gantt Chart is as follows(Figure 4).

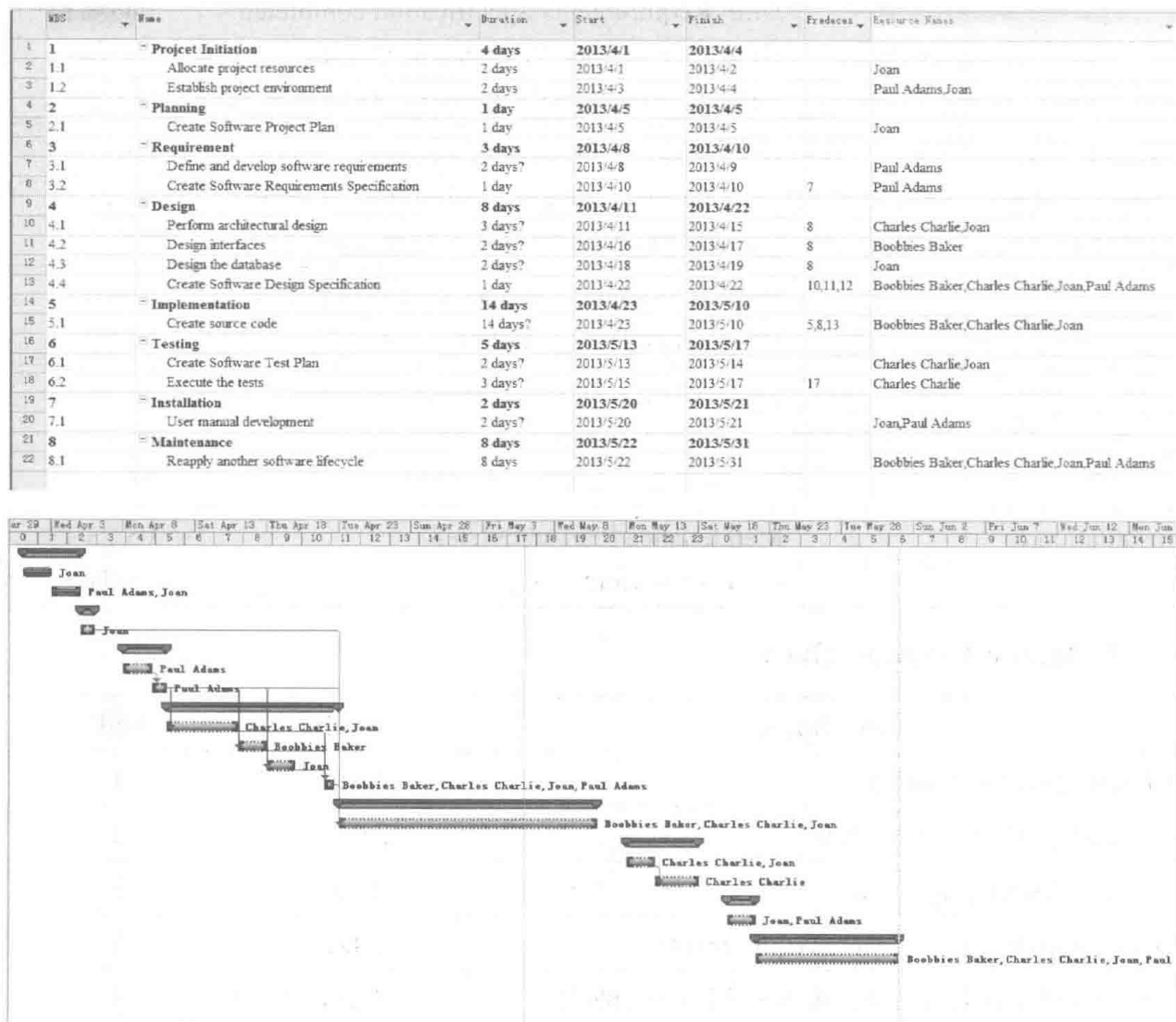


Figure 4 Gantt Chart

2.4 People(project group)/ Staff organization

Project Manager:

- Victor Lu

E-mail: Victor Lu@gmail.com

Experience: 2 years as a software developer at FB(part-time).

Familiar with programming in Java and C#.

Team:

- Jack Liang

E-mail: Jack Liang@hotmail.com

Experience: working in Expansion company as a product manager(part-time).

Familiar with prototype design.

- Jason Wang

E-mail: Jason Wang@gmail.com

Experience: working as a UI developer in Giant for 1 year(part-time).

Familiar with CSS, Javascript and HTML.

- Remonna Zeng

E-mail: RemonnaZeng@gmail.com

One year as a software developer at TC(part-time) .

Familiar with software engineering progress and programming in C++ and Java.

2.5 Management reporting and communication

Mechanisms for progress reporting and inter/intra team communication are identified.

Meetings

In order to monitor the progress of the project, the project group will be subjected to weekly meetings along with some informal meetings depending on need. During the weekly meetings, team members will discuss issues and progress through face-to-face meetings and all conclusions and decisions will be documented in the meeting reports. The informal meetings will be called throughout the entire project on an as-needed basis which will be determined by either team members or the clients.

E-mails

E-mails will be the daily communication channel for solving instant problems and confusions.

E-mails will also be used for scheduling the weekly meetings and transferring documents.

Status Report

Each week, every member has to fill a working-hour table for that week. The working-hour table separates the whole project into six stages. With the help of the working-hour table, both project manager(or team leader) and team members will be able to keep track of the time spent on the project.

3. Tracking and Control Mechanisms

Techniques to be used for project tracking and control are identified. In order to ensure the project goes well and the specifications are followed, the project team will set some monitoring policy.

3.1 Quality assurance and control

- Tight change management
- Quality review in a meeting format
- Extensive before implementation design using rapid prototyping
- Close contact with clients, meeting every two weeks and regular E-mail contacts

For complete information please refer to the document titled *Software Quality Assurance Plan*.

3.2 Change management and control

- For changes affect the user experiences we will have to notify all clients
- For changes that do not affect the user experiences we will notify a client representative
- Due to the size of the team, internal control panel will be used. One member of the team suggests a change, it will need to be approved by the other two members
- Formal version numbering will be used. All version changes must be documented in a common document accessible to all team members before a new version number can be released. Version number will be structured as follows:

<Major Release>.<Minor Release><Bug fix>

Version changes will be reviewed. Not only the previous version, but also all older versions of any document or codes will be preserved. This will ensure if the need to revert back to more than one version arises, the necessary version is available.

Change management policy

Once a work product has been finalized and approved, all changes to that work product must be submitted through the SVN, where the changes will be reviewed and either approved or denied by the project manager, based on the risk profile and perceived benefit of the change to be made. Since the waterfall methodology is being used for this project, requests for changes will be treated conservatively as they will potentially be extremely disruptive to the activities downstream of the change.

4. Method and Tool

Development methodology

The project shall use the waterfall software development methodology to deliver the software products, with work activities organized according to a tailored version of those provided by the IEEE Standard for Developing Software Life Cycle Processes (IEEE 1074-1997).

The Software Project Plan (SPP) shall be based on the IEEE Standard for Software Project Management Plans (IEEE 1058-1998).

Tools

The following work categories will have their work products satisfied by the identified tools:

Document Publishing:

- Microsoft Word 2013

Project management:

- Microsoft Project 2013

Implementation:

- Microsoft Visual Studio 2012
- Microsoft SQL Server 2012
- Adobe Dreamweaver CS 6

Version Control:

- Wandisco TortoiseSVN 1.7.9

5. Risk Management

The following part is the description of all the risks this project team might come up with.

- Team member leaves:

Team member may leave because of personal reasons.

How to avoid: Make sure every member understand the importance of the project and has the same determination to achieve it.

Minimize the risk: Make a backup plan, if the team member really leaves, the rest of the team can still complete the project. Before the team member is leaving, he/she should document all the working items he/she has done, so the other people can take it over.

- Static requirements :

The static requirements may be beyond the ability of the team member.

How to avoid: Check the requirements specification carefully, and do the preliminary analysis and feasibility study according to the requirements specifications. If there are technical difficulties, discuss with lecturer, team members and client representatives to find the solutions.

Minimize the risk: Discuss with lecturer, team members and client representatives to find the solutions.

- Develop a program that is not usable :

The resulted program may not fit the client's requirements or cannot run on the assigned platform.

How to avoid: Perform system analysis and system design carefully before writing the program. Do the checkpoints carefully.

Minimize the risk: Discuss with the lecturer, team members and client representatives, and see if this can be resolved by changing the design or implementation.

- Incompetent project management:

Incompetent project management may lead to failure of the project.

How to avoid: The project manager will remain up to date with the latest technology and keep track of the project development.

Minimize the risk: Discuss with the lecturer and team members and try to reorganize the team.

- Lack of knowledge :

Lack of technical and domain knowledge of the project.

How to avoid: All the team members have to study hard beforehand, and the client should provide learning resources or contact channels for the development team.

Minimize the risk: Ask for help from experts.

- Time related issues:

There may not be enough time for completing this project.

How to avoid: Conduct good time control and monitoring all the time.

Minimize the risk: Discuss with the lecturer and the client representatives, see if it's possible to either reduce the scale of the project or extend the developing time.

- Too ambitious requirements:

The client may ask for too many functions for this project.

How to avoid: Perform system analysis and system design carefully before writing the program. If there are too ambitious requirements, discuss them with the lecturer and the client representatives.

Minimize the risk: Discuss them with the lecturer and client representatives. Delay the implementing these requirements till next version or reduce the requirements.

- Bad design:

It can include: bad system design or user interface design or working flow design.

How to avoid: Analyze the system and the design properly at the beginning; consult the expert, lecturer and client representatives for their opinions. Review and integrate the system all the time.

Minimize the risk: Consult the experts and lecturer. If there is still enough time, redesign and re-implement the code.

- Lack of client interaction:

The project team is possible to lose the real time feedback from the client due to the lack of communication and interaction. This may lead to unusable or unqualified program.

How to avoid: Conduct review meetings and status reports regularly and make sure there is a contact channel between the project team and the client.

Minimize the risk: Visit the client regularly

- Lack of hardware resource:

The project team may not have enough hardware for developing this project, such as testing machines.

How to avoid: Prepare the hardware according to the system analysis.

Minimize the risk: Discuss this problem with the client and lecturer.

- Code theft :

The code may be stolen during the developing time.

How to avoid: Apply permission control to all the code written, use version control system to control the code, and all the code should have at least one backup.

Minimize the risk: Update the backup regularly and firewall the system.

Exercises

1. What is this plan mainly talking about?
2. What is the main constraint of this project?
3. What are the responsibilities of project manager in this project?

Working in a Team

团队合作

Part 1 Listening & Speaking

➤ Dialogue: Team Structure

Jason: Considering our situation, we can try Extreme Programming (XP) in the development process ^[1], which advocates communication without any obstacles and leads to a quick delivery. It is very appropriate to the characteristics of our team and project.

Sharon: You are right. With XP, we can cooperate with each other and focus our time and energy, which is needed to succeed.

Kevin: As I know, there is another team structure which recognizes that there are three main task categories in software development — management related, development related, and testing related. In this way, we can clearly divide our work into those three main aspects. Jason mainly focuses on development, Sharon conducts testing and I'm responsible for the management of our project, so that we can carry out the task more professionally.

Sharon: Sounds good! Well, let's do it!

Jason: Besides, in our daily work, good communication between members of a development group is essential. We must exchange information on the status of respective work, the design decisions that have been made and changes to previous decisions that are necessary.

Sharon: To accomplish this, we'd better establish some mechanisms for formal and informal communication among us. Formal communication can be accomplished through writing documents and progress reports. On the other hand, we can share ideas via Internet at anytime informally.

Kevin: In addition, since we develop synchronously and cooperatively,

[1] Replace with:
We can try an agile
programming in the
development process.

we need a configuration management mechanism in order to ensure the consistency of our work.

Jason: Yes. Maybe some automatic tools can help us with that.

Sharon: CVS, short for Concurrent Versions System, is one of the industry's most widely used version control systems^[2], and is very **fit for** the size and characteristics of a development team like ours. We can give it a try.

Kevin: Ok, I agree.

Replace with:
SVN, short for Subversion, is an open-source revision control system.

Words

advocate

['ædvəkit] v. 提倡, 主张

delivery

[di'livəri] n. 交付

synchronously

['sɪŋkrənəsli] adv. 同时地, 同步地

Phrases

fit for

适合

Abbreviations

XP	Extreme Programming	极限编程
CVS	Concurrent Versions System	并行版本系统, 一种代码版本控制软件
SVN	Subversion	一种代码版本管理软件

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

► Listening Comprehension: Project Team

Listen to the article and the 3 questions. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which point is not mentioned about the advantages of the team spirit for a group?
 - Observing a group quality standard self-consciously
 - Group members working closely together and learning from each other
 - Making every group members unique with a sense of "elite"
 - Practicing egoless programming

2. How many vivid terms describing the team spirit are referred in this article?
 (A) One
 (B) Two
 (C) Three
 (D) Not referred
3. Which is the most appropriate way of communication when making a most important change in design decisions?
 (A) The project manager makes the decision and informs other group members with an oral agreement.
 (B) Members working together share, negotiate and make the decision.
 (C) The system designer makes the decision and explains it to everyone involved in his daily report.
 (D) The whole group holds a meeting to discuss the decision, and write a formal report on it for the configuration management.

Words

motivate

['məʊtɪveɪt] v. 激发……的积极性

cohesiveness

[kəu'hi:sɪvnɪs] n. 黏结性, 内聚性

robust

[rə'bəʊst] adj. 强壮的, 精力充沛的

consensus

[kən'sensəs] n. (意见等) 一致

egoless

['i:gəʊlɪs] adj. 不自负的, 非自我的

jell

[dʒel] n. 果冻

knit

[nit] v. 接合

juggernaut

['dʒʌgənɔ:t] n. 骇人的(毁灭)力量

momentum

[məʊ'mentəm] n. 动力, 冲力, 势头

impersonal

[ɪm'pə:sənəl] n. 非个人的

ad hoc

['æd'hɒk] adv. 特别

Phrases

put together

把……放在一起

identify with

视……为一体, 认同

cope with

对付, 应付

go up

上升, 增长

➤ Dictation: Agile Software Development

This article will be played three times. Listen carefully, and fill in the blanks with the words you

have heard.

The modern business environment that **spawns** computer-based systems and software products is _____ 1 _____ and _____ 2 _____. Market conditions change rapidly, end-user needs _____ 3 _____, and new competitive threats _____ 4 _____ without warning. In many situations, we no longer are able to define requirements fully before the project begins. Software engineers must be agile _____ 5 _____ to respond to a fluid _____ 6 _____ environment.

The modern definition of agile software development evolved in the mid-1990s. _____ 7 _____, agile methods were called “_____ 8 _____ methods” as part of a reaction against “heavyweight” methods. The aim of it is to create software in a lighter, faster, more _____ 9 _____ way. In 2001, several of the most **prominent proponents** of those “lightweight methodologies” started the Agile _____ 10 _____ and signed the “**Manifesto** for Agile Software Development.”, a _____ 11 _____ of the values shared by them, for those **contemplating** new agile development processes, that is: individual and _____ 12 _____ over processes and tools, working software over comprehensive _____ 13 _____, _____ 14 _____ collaboration over contract negotiation, _____ 15 _____ to change over following a plan.

Actually, a number of methods similar to Agile were created _____ 16 _____ to 2000. _____ 17 _____ Programming (XP) is the most widely used agile process created by Kent Beck in 1996. An **adaptive** software development process was introduced in a paper by Edmonds in 1974. Other **notable** agile methods include Scrum, Crystal, _____ 18 _____ Software Development, _____ 19 _____ Driven Development, and _____ 20 _____ Systems Development Method.

Words

spawn

[spɔ:n] v. 造成, 大量产生

fluid

['flu:ɪd] adj. 变化的, 流动的, 液体的

prominent

['prɒmɪnənt] adj. 著名的

proponent

[prə'pəʊnənt] n. 支持者, 倡导者

manifesto

[.mæni'festəʊ] n. 宣言, 声明

contemplate

['kɒntempleɪt] v. 思考, 预期

adaptive

[ə'dæptɪv] adj. 适应的

notable

['nəʊtəbl] adj. 著名的

Part 2 Reading & Translating

➤ Section A: Team Structure

We have seen that the number of resources is **fixed** when schedule is being planned. Detailed scheduling is done only after actual assignment of people has been done, as task assignment

needs information about the capabilities of the team members. In general, we **implicitly** assume that the project's team is led by a project manager, who does the planning and task assignment. This form of hierarchical team organization is fairly common, and was earlier called the **Chief Programmer Team**.

In this hierarchical organization, the project manager is responsible for all major technical decisions of the project. He does most of the design and assigns coding of the different parts of the design to the programmers. The team typically consists of programmers, testers, a configuration controller, and possibly a librarian for documentation. There may be other roles like database manager, network manager, **backup** project manager, or a backup configuration controller. It should be noted that these are all logical roles and one person may do multiple such roles.

For a small project, a one-level hierarchy **suffices**. For larger projects, this organization can be extended easily by **partitioning** the project into modules, and having module leaders who are responsible for all tasks related to their module and having a team with them for performing these tasks. [1]

A different team organization is the egoless team. Egoless teams consist of ten or fewer programmers. The goals of the group are set by consensus, and input from every member is taken for major decisions. Group leadership **rotates** among the group members. Due to their nature, egoless teams are sometimes called democratic teams. This structure allows input from all members, which can lead to better decisions for difficult problems. This structure is well **suited for** long-term research-type projects that do not have time constraints. It is not suitable for regular tasks that have time constraints; for such tasks, the communication in democratic structure is unnecessary and **results in** inefficiency.

In recent times, for very large product developments, another structure has emerged.

This structure **recognizes** that there are three main task categories in software development—management related, development related, and testing related. It also recognizes that it is often desirable to have the test and development team be relatively independent, and also not to have the developers or testers report to a non-technical manager. [2] In this structure, consequently, there is an overall unit manager, under whom there are three small hierarchic organizations — for program management, for development, and for testing. The primary job of developers is to



write code and they work under a development manager. The responsibility of the testers is to test the code and they work under a test manager. The program manager provides the specifications for what is being built, and ensures that development and testing are properly coordinated. In a large product this structure may be **replicated**, one for each major unit. This type of team organization is used in corporations like Microsoft.

Words

fix

[fiks] v. 使固定, 确定

implicitly

[im'plisitli] adv. 不言明地, 默认地

hierarchical

[haɪə'rɑ:kikəl] adj. 分等级的, 层次的

backup

['bækʌp] adj. 后备的

suffice

[sə'faɪs] v. 足够, 满足

partition

[pɑ:'tiʃən] v. 划分, 把……分成部分

rotate

[rəu'teit] v. 轮流, 交替

recognize

['rekəgnaɪz] v. 认可, 承认

replicate

['replikɪt] v. 复制

Phrases

be suited for

适合于

result in

导致

Abbreviations

CPT Chief Programmer Team

首席程序员队

Complex Sentences

[1] **Original:** For larger projects, this organization can be extended easily by partitioning the project into modules, and having module leaders who are responsible for all tasks related to their module and having a team with them for performing these tasks.

Translation: 对于较大的项目, 通过将项目划分成模块, 并且拥有模块领导, 他们负责与其模块相关的所有任务并带领一支执行这些任务的团队, 这样的组织能够很容易地扩展。

[2] **Original:** This structure recognizes that it is often desirable to have the test and development team be relatively independent, and also not to have the developers or testers report to a non-technical manager.

Translation: 这种结构还认为, 让测试和开发团队相对独立, 以及不让开发人员或测试人员向非技术性经理汇报, 通常可以收到令人满意的效果。

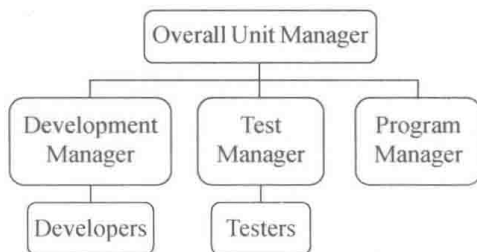
Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

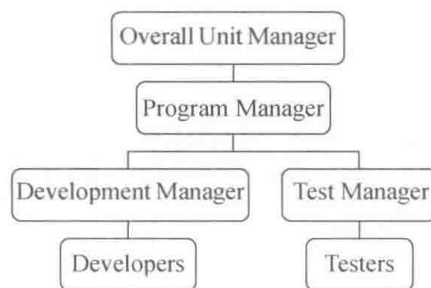
- ___ 1. Both the hierarchical team structure and the team structure used in Microsoft are extensible.
- ___ 2. Group leadership is not fixed in the egoless team.
- ___ 3. One person must act a specific and single role in the hierarchical team.
- ___ 4. Among all the team structures mentioned in this article, it is only in the Chief Programmer Team that a hierarchical structure is used.
- ___ 5. In general, there are three main task categories in software development.

II. Choose the best answer to each of the following questions.

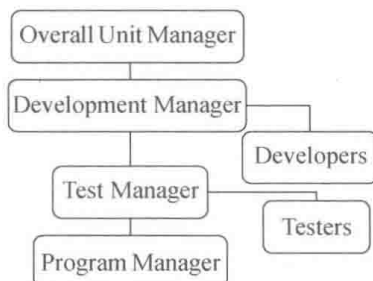
1. Which of the following statement is wrong according to this article?
 - (A) This article refers to three types of team structures in all.
 - (B) Hierarchical team organization is used the most frequently.
 - (C) Because of the project manager being responsible for all major technical decisions, the communication in the hierarchical organization is unnecessary and results in inefficiency.
 - (D) There is no the best team structure but only the most suitable one for a specific project.
2. Which of the following figures does properly present the third team structure mentioned in this article?



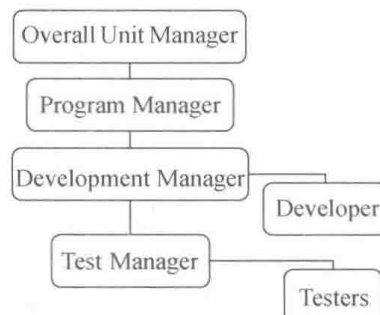
(A)



(B)



(C)



(D)

3. Suppose that you are going to develop a long-range exploratory project with some new techniques, which of the following team structures is the most suitable for you?
 - (A) One-level hierarchical team

- (B) Extended hierarchical team
- (C) Democratic team
- (D) The team organization which is used in Microsoft, mentioned in the last paragraph

III. Translation

1. Detailed scheduling is done only after actual assignment of people has been done, as task assignment needs information about the capabilities of the team members.
2. The goals of the group are set by consensus, and input from every member is taken for major decisions.

➤ Section B: The Internet of Things 物联网

No discussion of the future Internet would be complete without mentioning the Internet of Things(IoT), also sometimes referred to as the Industrial Internet.

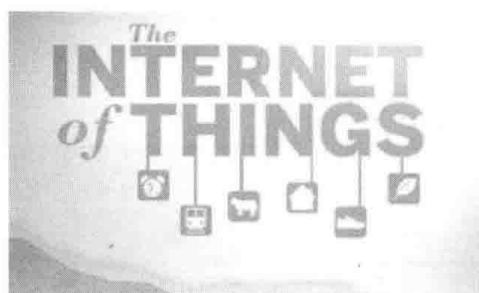
Internet technology is spreading beyond the desktop, laptop, tablet computer, and beyond the smartphone, to consumer electronics, electrical appliances, cars, medical devices, utility systems, machines of all types, even clothing - **just about** everything that can be equipped with sensors that collect data and connect to the Internet, enabling the data to be analyzed with data analytics software. The Internet of Things builds on foundation of existing technologies, such as RFID, and is being enabled by the availability of low cost sensors, the drop in price of data storage, the development of “Big Data” analytics software that can work with trillions of pieces of data, as well implementation of IPv6, which will allow Internet addresses to be assigned to all of these new devices. [1] Funding and research for the Internet of Things is being **spearheaded** by the European Union and China(where it is known as the Sensing Planet), and in the United States by companies such as IBM’s Smarter Planet initiative.

Although challenges remain before the Internet of Things is fully realized, it is coming closer and closer to **fruition**.

Today, the Internet of Things is in its early stages and is mostly used to monitoring things from **afar**. For instance:

- Ranchers are using wireless sensors on cattle to alert the rancher if a cow gets sick or lost.
- Wearable health tech lets doctors monitor patients with **chronic** illnesses.
- Sensors on household appliances can alert the manufacturer if an appliance needs maintenance or repair.

Eventually, the Internet of Things will do more than just alert you to the **whereabouts** or condition of something. Devices will talk to each other and grow autonomous. This is often called machine-to-machine technologies (M2M).



Examples include:

- Cars that drive themselves.
- Traffic lights that automatically respond to traffic jams or an accident, directing cars away from troubled areas.
- Buildings that automatically turn the lights and heat off when they detect no one is in the room.

Eventually, every item in the supply will be tracked and “smart”.

The whereabouts of coworkers will be instantly available, too.

“Employees on every level will be able to perform their functions more efficiently,” says W. David Stephenson, a consultant with INEX Advisors.

This will alter how managers run companies, getting rid of hierarchical decision makers — the classic role of the boss. “That, in my mind, will be the greatest business benefit of the Internet of Things,” Stephenson says.

As cool as the Internet of Things sounds, there is a **downside**. Issues such as privacy, reliability, and control of data still have to be **worked out**.

But even so, there’s no stopping the Internet of Things now. Within a few years, it will be like mobile phones and broadband — you won’t know how you ever lived without it.

Words

utility

[juːtɪləti] n. 公用事业, 效用

spearhead

[ˈspiəˈhed] v. 当……的先锋, 带头

fruition

[fruːɪʃən] n. 实现, 结果实

afar

[əˈfɑː] adv. 从远处

rancher

[ˈræntʃə] n. 大牧场主, 大农场工人

chronic

[ˈkrɒnɪk] adj. 慢性的, 长期的

whereabout

[ˈwerəˌbaʊt] n. 下落, 行踪, 所在之处

downside

[ˈdaʊnsaɪd] n. 缺点, 负面

issue

[ˈɪʃu] n. 问题

Phrases

just about

几乎, 几近, 大约, 马上

work out

解决, 算出

Abbreviations

IoT	Internet of Things	物联网
RFID	Radio Frequency Identification	射频识别技术, 俗称电子标签
IPv6	Internet Protocol version 6	因特网协议第 6 版

Complex Sentences

Original: The Internet of Things builds on foundation of existing technologies, such as RFID, and is being enabled by the availability of low cost sensors, the drop in price of data storage, the development of “Big Data” analytics software that can work with trillions of pieces of data, as well implementation of IPv6, which will allow Internet addresses to be assigned to all of these new devices.

Translation: 物联网建立在现有技术的基础上, 如射频识别技术 (RFID)。随着低成本传感器的可用性提高, 数据存储费用的降低, 可以处理万亿数据的大数据分析软件的发展, 以及允许为所有这些新设备分配互联网地址的 IPv6 技术的实现, 物联网逐渐成为可能。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. The Internet of Things (IoT) requires human-to-human or human-to-computer interaction.
- ___ 2. IPv6's huge increase in address space is an important factor in the development of the Internet of Things.
- ___ 3. If we had computers that knew everything there was to know about things — using data they gathered without any help from us — we would be able to track and count everything and greatly reduce waste, loss and cost.
- ___ 4. IoT is significant because an object that can represent itself digitally becomes something greater than when the object existed by itself.
- ___ 5. Issues such as privacy, reliability, and control of data have already been worked out by IoT.

II. Choose the best answer to each of the following questions.

1. Which statement is wrong about the Internet of Things?
 - (A) Issues such as privacy, reliability, and control of data still have to be worked out.
 - (B) Although challenges remain before the Internet of Things is fully realized, it is coming closer and closer to fruition.
 - (C) IPv4 is enough for the development of the Internet of Things.
 - (D) The Internet of Things will do more than just alert you to the whereabouts or condition of something.

2. Which of the following statements is wrong about funding and research for the Internet of Things being spearheaded by?
 - (A) China
 - (B) Russia
 - (C) United States
 - (D) European Union
3. Which of the following statements is wrong about the Internet of Things?
 - (A) It includes just about everything that can be equipped with sensors that collect data and connect to the Internet
 - (B) It includes just about everything that can be equipped without sensors that collect data and connect to the Internet
 - (C) It builds on foundation of existing technologies, such as RFID.
 - (D) It also sometimes referred to as the Industrial Internet.

III. Translation

Internet technology is spreading beyond the desktop, laptop, tablet computer, and beyond the smartphone, to consumer electronics, electrical appliances, cars, medical devices, utility systems, machines of all types, even clothing — just about everything that can be equipped with sensors that collect data and connect to the Internet, enabling the data to be analyzed with data analytics software.

Part 3 Writing: PowerPoint Presentation

Introduction

A PowerPoint presentation is similar to a poster presentation, only the information is on computer slides rather than actual posters. They are usually used to accompany an oral presentation; they should enhance the oral presentation. You can incorporate audio and visual media. They are often used to share information with a large group, such as at a professional conference, classroom presentations, and meetings. It should be more like an outline of your presentation. There are three main elements to a PowerPoint presentation: text, images, and tables or graphs. Text allows you to reinforce main points as well as key terms and concepts. Images illustrate or highlight main points. Tables and graphs present information in a way that is easy to understand and see.

Some Preliminaries

Before start, you should consider several key parts of your project: your audience, purpose (persuasive, informative, etc), subject matter and presentation. Since good PowerPoint projects come from making design decisions that fit the occasion, knowing as much as possible about the rhetorical situation before you create your PowerPoint will ensure your success.

Common Components

The slides for a PowerPoint presentation should be more like an outline. Text is often listed rather than written in full sentences. The following are a few of the things that can be presented on a PowerPoint slide:

- Graphs and/or tables
- Definitions
- Lists
- Essential facts
- Necessary images

Arrangement

The order of your PowerPoint will depend on what you think your audience needs. Whatever you do, organize your PowerPoint carefully and present your argument methodically so that your audience bury into your argument. There are some options that can be referred to.

- Overview, Body, Conclusion (In the case that your audience need clarity in your presentation because you think there is a risk of them getting lost in your complicated points).
- Anecdote, Content, Conclusion (In the case that your audience might be bored even before you start).
- Plan, Benefits, Anecdote (In the case that you are attempting to convince an audience to establish something new and are facing an audience that demands your presentation be short and to the point).

Other Useful Information

Do:

- Choose a single background for the entire presentation.
- Use simple, clean fonts.
- Use a font size that can be seen from the back of the room.
- Write in bulleted format and use consistent phrase structure in lists.
- Provide essential information only. Use key words to guide the reader/listener through the presentation.
- Use direct, concise language. Keep text to a minimum.
- Provide definitions when necessary.
- Use white space to set off text and/or visual components.
- Make sure each slide logically leads to the next.
- Use a heading for each slide.

Don't:

- Clutter the slide with graphics.
- Use complicated fonts.

- Add superfluous information.
- Put down every word you are going to say.
- Use images if they will distract.
- Use hard to read color combinations, like black on blue. Try to use high contrast combinations.

A Sample PowerPoint Presentation

Introduction to Extreme Programming (XP)

Source: ExtremeProgramming.org/home

1

What is XP

- Extreme Programming (XP) is actually a deliberate and disciplined approach to software development. It was based on observations of what made computer programming faster and what made it slower. About eight years old, it has already been proven at many companies of all different sizes and industries world wide.
- XP is an important new methodology for two reasons:
 - it is a re-examination of software development practices that have become standard operating procedures
 - It is one of several new lightweight software methodologies created to reduce the cost of software.
- XP is successful because it emphasizes:
 - Customer involvement and satisfaction
 - Team work
- XP improves a software project in four essential ways:
 - Communication
 - Simplicity
 - Feedback
 - Courage

2

When to use XP

- Dynamic requirements
- High project risks
- Small groups of programmers
- Testability
- Productivity

3

The Rules and Practices of XP

■ Planning

- » User stories are written.
- » Release planning creates the schedule.
- » Make frequent small releases.
- » The Project Velocity is measured.
- » The project is divided into iterations.
- » Iteration planning starts each iteration.
- » Move people around.
- » A stand-up meeting starts each day.
- » Fix XP when it breaks.

■ Designing

- » Simplicity.
- » Choose a system metaphor.
- » Use CRC cards for design sessions.
- » Create spike solutions to reduce risk.
- » No functionality is added early.
- » Refactor whenever and wherever possible.

4

The Rules and Practices of XP (cont.)

■ Coding

- » The customer is always available.
- » Code must be written to agreed standards.
- » Code the unit test first.
- » All production code is pair programmed.
- » Only one pair integrates code at a time.
- » Integrate often.
- » Use collective code ownership.
- » Leave optimization till last.
- » No overtime.

■ Testing

- » All code must have unit tests.
- » All code must pass all unit tests before it can be released.
- » Acceptance tests are run often and the score is published.

5

XP Map

- The XP Map shows how they work together to form a development methodology. Unproductive activities have been trimmed to reduce costs and frustration.

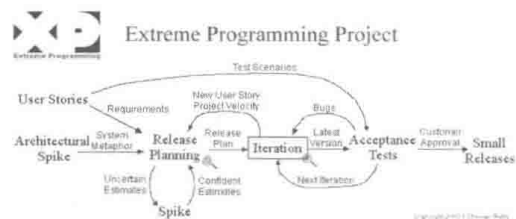


Figure 1 XP Map

6

What We Have Learned About XP

- Release Planning
- Simplicity
- System Metaphor
- Pair Programming
- Integrate Often
- Optimize Last
- Unit Tests
- Acceptance Tests

7

More Information

- Web Sites
 - The Portland Pattern Repository
 - XProgramming.com
 - XP Developer
 - ...
- Books
 - *Extreme Programming Explained: Embrace Change*. By Kent Beck
 - *Refactoring Improving the Design of Existing Code*. By Martin Fowler
 - *Extreme Programming Installed*. By Ron Jeffries, Chet Hendrickson, and Ann Anderson
 - ...

8

Exercises

1. Write a similar presentation regarding the Introduction to Software Requirement.
2. Write a similar presentation regarding the Introduction to Software Project Plan.

Designing the System

系统设计

Part 1 Listening & Speaking

➤ Dialogue: Software Design

Sharon: Having completed the analysis modeling and requirements specification of the system, we can progress further!

Kevin: Yes, now let's have a discussion on our generic task set in the design process.

Jason: We will move from a "big picture" view of the software to a more narrow view that defines the details required to implement this system.^[1] The process begins by focusing on architecture, I think.

Sharon: What do you mean?

Jason: Using the analysis model we constructed last week, we need to select an architectural style that is appropriate for our software.

Kevin: After that, we can partition the analysis model into design subsystems and allocate these subsystems within the architecture.

Jason: In the detailed design phase, we must be certain that^[2] each subsystem is functionally cohesive, and design subsystem interfaces first, and then allocate analysis classes or functions to each subsystem.

Kevin: Additionally, it is an important factor for successful software to have a user interface **appreciated** by its users, particularly for business software like ours.

Jason: Sharon is a **competent** art designer.

Sharon: I'll do my best.

Kevin: We will collaborate to specify action sequences based on user scenarios and help Sharon to create a behavioral model of the interface.

Sharon: Ok, next, I will define interface objects and control mechanisms

[1] Replace with:
We perform the software design from a high level to more detailed levels step by step.

[2] Replace with:
1. We must ensure that...
2. We must make sure that...

and **beg** you to review and revise as required.

Kevin: No problem. After completing the architecture design, Jason and I will conduct component-level design by specifying all algorithms at a relatively low level of abstraction, define component level data structure, and refine the interface of each component if necessary

Jason: After reviewing the component level design, we can develop a deployment model for our system.

Words

view

[vju:] n. 视图, 观点

appreciate

[ə'pri:ʃieit] v. 赏识, 欣赏, 重视

competent

['kɒmpitənt] adj. 有能力的, 胜任的

beg

[beg] v. 请求, 恳求

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ Listening Comprehension: Software Design

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which description is not correct about the design process?
 - The design process moves from a macro-view of software to a more microcosmic view.
 - Through the design process, requirements are translated into a “blueprint” for construction.
 - The design process is represented from a high level of abstraction to lower levels step by step.
 - It is reasonable for the design process to create a model of software with all details directly.
- How many different elements are encompassed in the design model according to this article?
 - Two
 - Three
 - Four
 - Five
- Which statement is not correct about the design concepts?

- (A) Design concepts have been developing over the first half-century of software engineering work.
- (B) The design concepts used in different projects may be inconsistent because of the different software engineering process that is chosen.
- (C) The design methods applied in the projects may be different in terms of the different software engineering process that is chosen.
- (D) Design concepts have evolved along with software engineering.

Words

commence

[kə'mens] n. 开始

iteration

[,itə'reiʃən] n. 迭代, 反复

sift

[sift] v. 详审, 精选

converge

[kən'veɪdʒ] v. 会聚

encompass

[in'kʌmpəs] v. 包含, 包括

populate

['pɒpjuleit] v. 构成, 填充

Phrases

come to a conclusion

得出结论, 告一段落

➤ Dictation: User Interface Design

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

When the concept of the interface first began to emerge, it was commonly understood as the hardware and software through which a human and a computer could communicate. So User Interface Design is also known as ____1____ Interaction or HCI. As it has evolved, the concept has come to include the **cognitive** and emotional aspects of the user's ____2____ as well.

Many technological innovations ____3____ upon User Interface Design to **elevate** their technical complexity to a ____4____ product. Technology alone may not win user ____5____ and subsequent ____6____. The User Experience, or how the user experiences the end product, is the key to acceptance. And that is where User Interface Design enters the design ____7____. While product engineers ____8____ on the technology, usability ____9____ focus on the UI. For greatest efficiency and cost effectiveness, this working ____10____ should be maintained from the start of a project to its rollout.

UI affects the feelings, the emotions, and the mood of your users. If the UI is wrong and the user feels like they can't ____11____ your software, they **literally** will be angry and ____12____ it

on your software. 13 , if the UI is smart and things work the way the user expected them to work, they will be cheerful as they manage to 14 small goals. Thus, the **cardinal axiom** of all user interface design is that, a UI is 15 when the program 16 exactly how the user thought it would.

To make user happy, you have to 17 them in control of their environment. To do this, you need to correctly interpret their actions. Additionally, you should reduce the user's memory 18 and make the interface 19 . Those three points above are considered as "20" and actually form the basis for a set of user interface design principles that guide this important software design action.

Words

cognitive

['kɒgnitiv] adj. 认知的, 认识的

elevate

['eliveit] v. 提高, 提升

rollout

['rəʊl.aʊt] n. 首次展示, (产品或服务
的) 正式推出

literally

['lɪtərəli] adv. 真实地, 确切地

cardinal

['kɑ:dɪnəl] adj. 主要的, 最重要的

axiom

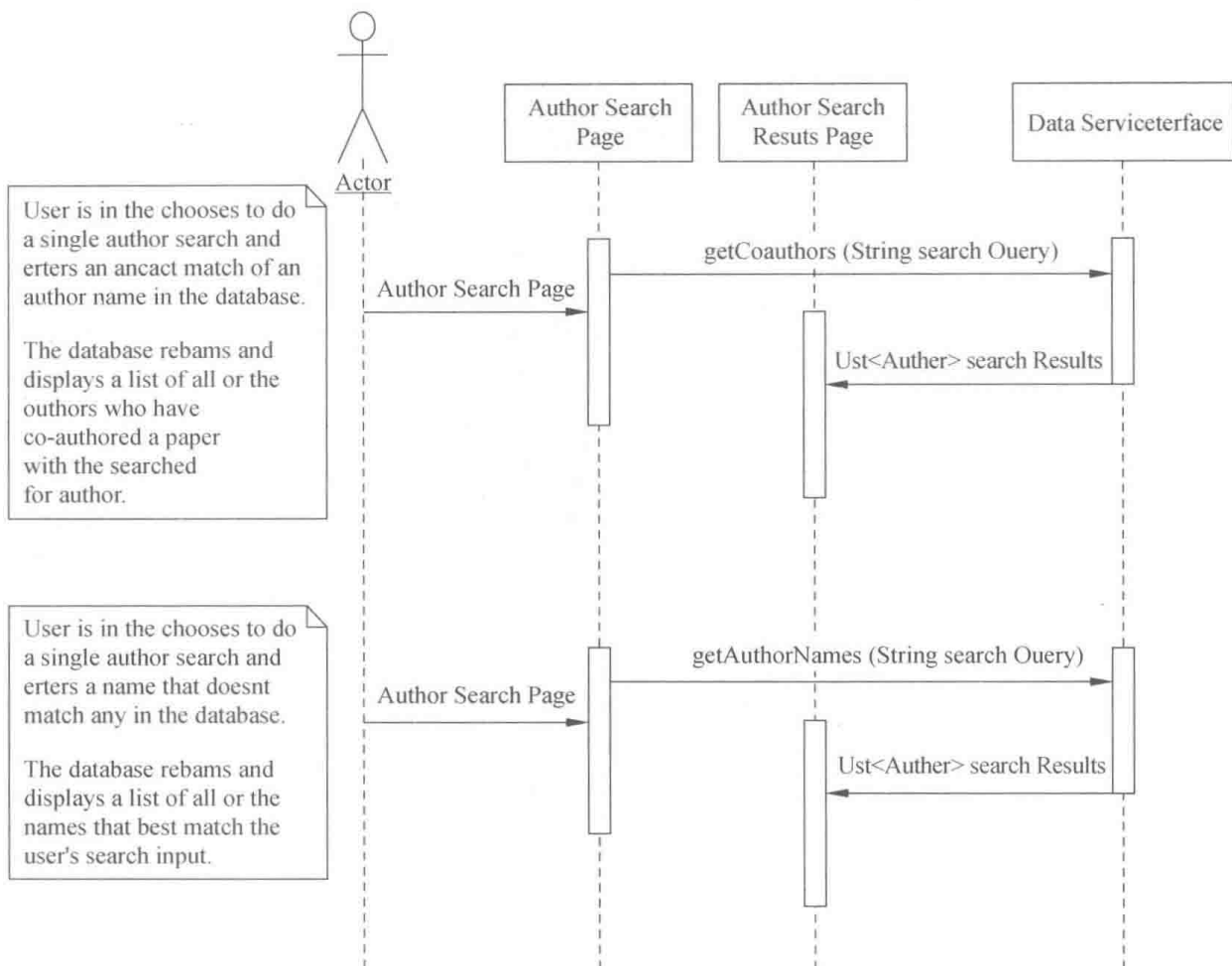
['æksiəm] n. 公理, 格言, 规则

Part 2 Reading & Translating

➤ Section A: Software Design

Design is defined in [IEEE610.12-90] as both "the process of defining the architecture, components, interfaces, and other characteristics of a system or component" and "the result of that process." Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will **serve as** the basis for its construction. [1] More precisely, a software design (the result) must describe the software architecture — that is how software is decomposed and organized into components — and the interfaces between those components. It must also describe the components at a level of detail that enable their construction. [2]

Software design plays an important role in developing software: it allows software engineers to produce various models that form a kind of **blueprint** of the solution to be implemented. We can analyze and evaluate these models to determine whether or not they will allow us to fulfill the various requirements. We can also examine and evaluate various alternative solutions and **trade-offs**. Finally, we can use the resulting models to plan the subsequent development activities, in addition to using them as input and the starting point of construction and testing.



In a standard listing of software life cycle processes such as IEEE/EIA 12207 Software Life Cycle Processes [IEEE 12207.0-96], software design consists of two activities that fit between software requirements analysis and software construction:

- Software architectural design (sometimes called top level design): describing software's top-level structure and organization and identifying the various components
- Software detailed design: describing each component sufficiently to allow for its construction.

General Strategies

Some often-cited examples of general strategies useful in the design process are **divide-and-conquer** and **stepwise** refinement, top-down vs. bottom-up strategies, data abstraction and information hiding, use of heuristics, use of patterns and **pattern languages**, use of an iterative **and incremental approach**.

Function-Oriented (Structured) Design

This is one of the classical methods of software design, where decomposition centers on identifying the major software functions and then elaborating and refining them in a top-down manner. Structured design is generally used after structured analysis, thus producing, **among other things**, data flow diagrams and associated process descriptions. Researchers have proposed

various strategies (for example, transformation analysis, **transaction** analysis) and heuristics (for example, **fan-in/fan-out**, scope of effect vs. scope of control) to transform a DFD into a software architecture generally represented as a structure chart.

Object-Oriented Design

Numerous software design methods based on objects have been proposed. The field has evolved from the early object-based design of the mid-1980s (noun=object; verb=method; adjective=attribute) through OO design, where **inheritance** and **polymorphism** play a key role, to the field of component-based design, where **meta-information** can be defined and accessed (through reflection, for example). Although OO design's roots **stem from** the concept of data abstraction, responsibility-driven design has also been proposed as an alternative approach to OO design.

Data-Structure-Centered Design

Data-structure-centered design (for example, Jackson, Warnier-Orr) starts from the data structures a program manipulates rather than from the function it performs. The software engineer first describes the input and output data structures (using **Jackson's structure diagrams**, for instance) and then develops the program's control structure based on these data structure diagrams. Various heuristics have been proposed to deal with special cases — for example, when there is a mismatch between the input and output structures.

Component-Based Design (CBD)

A software component is an independent unit, having **well-defined** interfaces and dependencies that can be composed and deployed independently. Component-based design addresses issues related to providing, developing, and integrating such components in order to improve **reuse**.

Other Methods

Other interesting but less mainstream approaches also exist: formal and rigorous methods and transformational methods.

Words

blueprint

['blu:ˌprɪnt] n. 蓝图, 计划

trade-off

[treɪdɔf] n. 平衡, 折中

cite

[saɪt] v. 引用

stepwise

[stepwaɪz] adj. 逐步的

heuristic

[hjuəˈrɪstɪk] adj. & n. 启发式的, 启发式的论据

iterative

['ɪtəreɪtɪv] n. & adj. 迭代 (的)

transaction

[trænˈzækʃən] n. 事务

fan-in

['fænɪn] n. 扇入, 输入端

Words**fan-out**

['fæ,aut] n. 扇出, 输出端

inheritance

[in'heritəns] n. 继承

polymorphism

[ˌpɒli'mɔːfɪzəm] n. 多态

stem

[stem] v. 起源于

well-defined

['weldi'faɪnd] adj. 明确的

reuse

['ri:'ju:z] n. 复用

rigorous

['rɪgərəs] adj. 精确的, 严密的

Phrases

serve as

充当, 用作

divide-and-conquer

分而治之

pattern language

模式语言

iterative and incremental approach

迭代和增量方法

among other things

其中

meta-information

元信息

stem from

基于, 处于

Jackson's structure diagram

Jackson 结构图

Abbreviations

EIA	Electronic Industries Association	美国电子工业协会
DFD	Data Flow Diagram	数据流图
OO	Object-Oriented	面向对象的
CBD	Component-Based Design	基于组件的设计

Complex Sentences

[1] Original: Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction.

Translation: 作为一个过程来看, 软件设计是一项软件工程生命周期活动。其中, 软件需求被分析以产生软件内部结构的描述, 将作为软件构建的基础。

[2] Original: More precisely, a software design (the result) must describe the software architecture—that is how software is decomposed and organized into components — and the interfaces between those components. It must also describe the components at a

level of detail that enable their construction.

Translation: 更确切地讲, 软件设计(的结果)必须描述软件体系结构——即软件如何被分解和组织成组件, 以及这些组件之间的接口。软件设计也必须在使其能够构建的详细级别上描述组件。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. In software design, software engineers form a blueprint of the solution to be implemented.
- ___ 2. Structure Chart is a kind of diagram for representing the software architecture in Function-Oriented Design.
- ___ 3. Object-Oriented Design has the same meaning but a different name with object-based design emerged in the mid-1980s.
- ___ 4. Inheritance and polymorphism are two of the most key characteristics in OO design.
- ___ 5. A software component has well-defined interfaces and dependencies that can be composed and deployed independently.

II. Choose the best answer to each of the following questions.

- 1. Which statement is wrong about the software design according to this article?
 - (A) Software design plays an important role in transition from software requirements to its construction.
 - (B) Software design describes the software architecture, interfaces and components, and defines other characteristics of a system or component.
 - (C) Using heuristics and an iterative approach as general strategies, software design is a heuristic and iterative process in nature.
 - (D) Software design allows software engineers to analyze software requirements and form a model of the requirements finally.
- 2. Which statement is right about the software design approaches mentioned in this article?
 - (A) Structured design focuses on identifying the major software structure and then elaborating and refining them in a top-down manner.
 - (B) In Data-Structure-Centered Design, the software engineer first produces the input and output DFDs and then develops the program's control structure.
 - (C) In Object-Oriented Design, every noun can be strictly abstracted into an object, and every verb can be abstracted into a method as well as every adjective can be abstracted into an attribute of the object related.
 - (D) The main objective of Component-Based Design is improving software reuse.
- 3. How many kinds of methods for software design are referred to in this article in all?
 - (A) Five
 - (B) Six

(C) Seven

(D) Eight

III. Translation

Data-structure-centered design (for example, Jackson, Warnier-Orr) starts from the data structures a program manipulates rather than from the function it performs.

➤ Section B: Social Networking 社交网络

Social networking is the grouping of individuals into specific groups, like small rural communities or a **neighborhood** subdivision, if you will. Although social networking is possible in person, especially in the workplace, universities, and high schools, it is most popular online.

This is because unlike most high schools, colleges, or workplaces, the Internet is filled with millions of individuals who are looking to meet other people, to gather and share first-hand information and experiences about cooking, **golfing**, gardening, developing friendships, professional alliances, finding employment, business-to-business marketing and even groups sharing information about baking cookies to the **Thrive Movement**. [1] The topics and interests are as varied and rich as the story of our universe.

When it comes to online social networking, Websites are commonly used. These Websites are known as social sites. Social networking Websites function like an online community of Internet users. Depending on the Website **in question**, many of these online community members share common interests in hobbies, religion, politics and alternative lifestyles. Once you are granted access to a social networking Website you can begin to socialize. This socialization may include reading the **profile** pages of other members and possibly even contacting them.

The friends that you can make are just one of the many benefits to social networking online. Another one of those benefits includes diversity because the Internet gives individuals from all around the world access to social networking sites. This means that although you are in the United States, you could develop an online friendship with someone in Denmark or India. Not only will you make new friends, but also you just might learn a thing or two about new cultures or new languages and learning is always a good thing.

As mentioned, social networking often involves grouping specific individuals or organizations together. While there are a number of social networking Websites that focus on particular interests, there are others that do not. The Websites without a main focus are often referred to as “traditional” social networking Websites and usually have open memberships. This means that anyone can become a member, no matter what their hobbies, beliefs, or views are. However, once you are inside this online community, you can begin to create your own network of friends and eliminate members that do not share common interests or goals.



As we know, there are dangers associated with social networking including data theft and viruses, which are on the rise. The most **prevalent** danger though often involves online **predators** or individuals who claim to be someone that they are not. Although danger does exist with networking online, it also exists in the real world, too. Just like you're **advised** when meeting strangers at clubs and bars, school, or work --you are also advised to proceed with caution online.

By being aware of your cyber-surroundings and who you are talking to, you should be able to safely enjoy social networking online. It will take many phone conversations to get to know someone, but you really won't be able to make a clear judgment until you can meet each other in person. Just use **common sense** and listen to your inner voice; it will tell you when something doesn't feel right about the online conversations taking place.

Once you are well **informed** and comfortable with your findings, you can begin your search from hundreds of networking communities to join. This can easily be done by performing a standard Internet search. Your search will likely return a number of results, including MySpace, FriendWise, FriendFinder, Yahoo! 360, Facebook, Orkut, and Classmates.

Words

subdivision

[ˌsʌbdɪˈvɪʒn] n. 细分, 一部, 分支

golf

[gɒlf] v. 打高尔夫球

profile

[ˈprəʊfaɪl] n. 人物简介, 轮廓, 侧面

prevalent

[ˈprevlənt] adj. 流行的, 盛行的, 普遍存在的, 普遍发生的

predator

[ˈpredətə] n. 以掠夺为生的人, 食肉动物

advise

[ədˈvaɪz] v. 忠告, 给……出主意, 劝告

inform

[ɪnˈfɔrm] v. 通知, 告知

Phrases

neighborhood subdivision	邻域细分
Thrive Movement	发展运动
in question	考虑之中的, 被谈论着的
common sense	常识 (尤指判断力)

Complex Sentences

Original: This is because unlike most high schools, colleges, or workplaces, the Internet is filled with millions of individuals who are looking to meet other people, to gather and share first-hand information and experiences about cooking, golfing, gardening, developing friendships, professional alliances, finding employment, business-to-business marketing and even groups sharing information about baking cookies to the Thrive Movement.

Translation: 这是因为, 与大部分中学、大学或工作场所不同, 互联网上有着数以百万计的个人用户。他们在不断地尝试和别人建立联系, 以获取或分享关于烹饪、打高尔夫球、园艺、发展友谊、职业联盟、求职、企业对企业营销甚至互联网上还有有关为“发展运动”烘制小饼干分享信息的群的第一手信息和经验。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. When it comes to online social networking, FTP is commonly used.
- ___ 2. Social networking Websites function like an auction between Internet users.
- ___ 3. Facebook and Twitter are examples of social networking options.
- ___ 4. Social networking sites allow users to share ideas, pictures, posts, activities, events, and interests with people in their network.
- ___ 5. Regarding data theft and viruses, it is not dangerous to use social networking service.

II. Choose the best answer to each of the following questions.

1. Which statement is wrong about the social networking?
 - (A) Although danger does exist with social networking online, it also exists in the real world, too.
 - (B) Social networking often involves grouping specific individuals or organizations together.
 - (C) Making friends is just one of the many benefits to social networking online.
 - (D) Data theft and viruses have no threats to social networking online.
2. Which company is social networking related?
 - (A) Facebook

- (B) Intel
 - (C) Dell
 - (D) Lenovo
3. Which of the following is not the work of social networking Websites?
- (A) Social networking Websites function like an online community of Internet users.
 - (B) Many of these online community members share common interests in hobbies, religion, politics and alternative lifestyles via social networking Websites.
 - (C) Once you are granted access to a social networking Website you are allowed to read the profile pages of other members and possibly even contacting them.
 - (D) Once you are granted access to a social networking Website you can begin to individualize, and need no contacts with others.

III. Translation

1. Another one of those benefits includes diversity because the Internet gives individuals from all around the world access to social networking sites.
2. The most prevalent danger though often involves online predators or individuals who claim to be someone that they are not.

Part 3 Simulated Writing: Software Design Specification

An important product of the design process is a set of documents that describe the system to be built. As we have seen, one part must tell the customers and users in natural language what the system will do; a second part uses technical terminology to describe the system's structure, data, and functions. Thus, the contents of the two parts may overlap, but the ways of expressing them may not.

The design documents should contain a section, called the design rationale, outlining the critical issues and trade-offs that were considered in generating the design. This guiding philosophy helps the customers and other developers understand how and why certain parts of the design fit together.

The design also contains descriptions of the components of the system. One section should address how the users interact with the system.

Usually, a set of diagrams or formal notations describe the overall organization and structure of the system, including all levels of abstraction.

If the system is distributed, the configuration in the design is detailed enough to show the topology of the network, how the network nodes will access one another, and the allocation of functions to the nodes. If the system requirements include constraints on timing, or if the nodes

of the network must be synchronized, the design describes how the timing will work. Similarly, the design contains information about the control and routing of messages. It may also include prescriptions for the integrity of the network: making sure that the data are accurate or can be recovered after a failure.

If the customer requires it, elements of the design may address monitoring system performance. In addition, there may be a manual override of the system, and the design describes how it will work. Other sections of the design documents may address fault location and isolation, system reconfiguration, or special security measures.

Finally, the design is cross-referenced with the requirements to demonstrate how the design is derived from them. This correspondence forces us to check for completeness and consistency. In addition, such a cross-reference will make enhancements or modifications easier to track later. For example, if a requirement changes, the cross-reference points to the corresponding design changes needed.

Software Design Specification Template

1. Introduction

This section (1-2 pages) provides an overview of this entire document.

1.1 Project Overview

Describe the client, the problem to be solved, and the intended users. Explain the context in which your software will be used, i.e. the big picture(1-3 paragraphs).

1.2 Project Scope

Mention the most important features of the system, inputs, data stores, and outputs. Do not discuss implementation details. Note any major constraints(1-5 paragraphs).

1.3 Document Preview

Describe the purpose, scope of this document, and intended audience of this document. Mention the major sections that follow. Provide references to companion documents.(1-2 paragraphs).

2. Architectural Design

This section (2-4 pages) provides an overview and rationale for the program's data and architectural design decisions.

2.1 Section Overview

Provide a summary of the contents of this section (1-2 paragraphs).

2.2 General Constraints

Describe global limitations or constraints that have a significant impact on your system design. Examples include hardware and software environments, interface requirements, external data representations, performance requirements, network requirements, etc(1-3 paragraphs).

2.3 Data Design

Describe the structure of any databases, external files, and internal data structures. You may wish to include references to appendices containing ER diagrams, data, or file formats (1-3 paragraphs).

2.4 Program Structure

Describe the architectural model chosen and the major components. Include a pictorial representation (or reference to an appendix block or class diagram) of the major components (1-4 paragraphs).

2.5 Alternatives Considered

Discuss the alternative architectural models considered and justify your choice for your architectural design (1-4 paragraphs).

3. Detailed Design

This section represents the meat of your document. Be as detailed as time allows.

3.1 Section Overview

Provide a summary of the contents of this section (1-2 paragraphs).

3.2 Component *n* Detail (include a sub-section for each component)

A structured description usually works. For example, if your components are classes you may wish to include the following subsections

3.2.1 Description

3.2.2 Data Members (include type, visibility, and description)

3.2.3 Methods (include English or pseudocode descriptions for each one)

4. User Interface Design

4.1 Section Overview

Provide a summary of the contents of this section (1-2 paragraphs).

4.2 Interface Design Rules

Describe and justify the conventions and standards used to design your interface. (1-2 paragraphs).

4.3 GUI Components

Note the GUI components or API's provided in the development environment that you plan on using (1 paragraph + table).

4.4 Detailed Description

Provide a detailed description of the user interface including screen images. You may prefer to reference an appendix containing the screen snapshots (1-4 pages).

5. Conclusion

Provide an ending to this document with a mention of implementation and testing strategies resulting from this design (1-2 paragraphs).

6. Appendices (a list of possibilities)

6.1 Database Entity-Relationship Diagram

6.2 Architectural Design Block Diagram(s)

6.3 Class Diagram(s)

6.4 Class Sequence Diagram(s)

6.5 User Interface Screen Snapshots

PS: This template can be tailored according to the different projects.

A Sample Software Design Specification

Software Design Specification For Web Publishing System Version 1.0

Victor Lu (Team Leader)

Jack Liang

Jason Wang

Remonna Zeng

April 22, 2013

 *Table of Contents***TABLE OF CONTENTS**

- 1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Glossary
 - 1.4 References
 - 1.5 Overview of Document
- 2. Deployment Diagram
- 3. Architectural Design
 - 3.1 On-Line Journal
 - 3.1.1 On-Line Journal
 - 3.1.2 Author Search Form
 - 3.1.3 Category Search Form
 - 3.1.4 Keyword Search Form
 - 3.1.5 Article Table
 - 3.1.6 Author Table
 - 3.1.7 Category Table
 - 3.2 Article Manager
 - 3.2.1 People Manager Form
 - 3.2.2 Article Manager Form
 - 3.2.3 Publisher Form
 - 3.2.4 Historical Society Interface
 - 3.2.5 Author Table
 - 3.2.6 Reviewer Table
 - 3.2.7 Active Article Table
 - 3.2.8 Author Relationship Table
 - 3.2.9 Reviewer Relationship Table
 - 3.2.10 Category Relationship Table
 - 3.2.11 Article Table
- 4. Data Structure Design
 - 4.1 On-Line Journal Database
 - 4.1.1 Article Table
 - 4.1.2 Author Table
 - 4.1.3 Category Table
 - 4.2 Article Manager Database
 - 4.2.1 Author Table

- 4.2.2 Reviewer Table
- 4.2.3 Active Article Table
- 4.2.4 Author Relationship Table
- 4.2.5 Reviewer Relationship Table
- 4.2.6 Category Relationship Table
- 5. User Interface Design
 - 5.1 On-Line Journal User Interface
 - 5.2 Article Manager User Interface
- 6. Real-Time Design
- 7. Help System Design
- 8. Use Case Realizations
 - 8.1a Search Article – Author
 - 8.1.b Search Article-Category
 - 8.1c Search Article – Keyword
 - 8.2 Communicate
 - 8.3 Add Author
 - 8.4 Add Reviewer
 - 8.5a Update Person – Author
 - 8.5b Update Person – Reviewer
 - 8.6a Update Article Status – Delete Reviewer
 - 8.6b Update Article Status – Add/Delete Category
 - 8.7 Enter Communication
 - 8.8 Assign Reviewer
 - 8.9 Check Status
 - 8.10 Send Communication
 - 8.11 Publish Article
 - 8.12 Remove Article

Index

 List of Figures

Figure 1 Deployment Diagram

Figure 2 Top-Level Architectural Design

Figure 3 On-Line Journal Architectural Design

Figure 4 Article Manager Architectural Design

Figure 5 Sequence Diagram of Search Article by Author

Figure 6 Sequence Diagram of Search Article by Category

Figure 7 Sequence Diagram of Search Article by Keyword

Figure 8 Sequence Diagram of Communicate

Figure 9 Sequence Diagram of Add Author

Figure 10 Sequence Diagram of Add Reviewer

Figure 11 Sequence Diagram of Update Person by Author

Figure 12 Sequence Diagram of Update Person by Reviewer

Figure 13 Sequence Diagram of Update Article Status by Delete Reviewer

Figure 14 Sequence Diagram of Update Article Status by Add /Delete Category

Figure 15 Sequence Diagram of Assign Reviewer

Figure 16 Sequence Diagram of Send Communication

Figure 17 Sequence Diagram of Publish Article

Figure 18 Sequence Diagram of Remove Article

1. Introduction

1.1 Purpose

This document contains the complete design description of the *Web Publishing System*. This includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the SRS will be implemented in the system using this design.

The primary audiences of this document are the software developers.

1.2 Scope

There are two parts of this system. The On-Line Journal uses standard client-server architecture with a database on the server. The Article Manager is a repository architecture using a database. This system does not interface with any external system although the Editor will use E-mails with attached files outside of the system. These files will be brought into the system or will be exported to files which will be attached to E-mails. <<Since we identified Domain (Analysis) classes in the SRS, we are utilizing them in the design >>.

1.3 Glossary

Term	Definition
Active Article	The document that is tracked by the system; it is a narrative that is planned to be posted to the public Website
Author	Person submitting an article to be reviewed. In case of multiple authors, this term refers to the <i>principal author</i> , with whom all communication is made
Editor	Person who receives articles, sends articles for review, and makes final judgments for publications
Historical Society Database	The existing membership database (also HS database)
Keyword	Word or phrase used in searching by keyword. Case is not significant
Member	A member of the Historical Society listed in the HS database
Reader	Anyone visiting the site to read articles
Review	A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement
Reviewer	A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made in the article
User	Reviewer or Author

1.4 References

Software Requirements Specification for Web Publishing System, 2013.

1.5 Overview of Document

- Chapter 2 is a Deployment Diagram that shows the physical nodes on which the system resides. This allows a clear explanation of where each design entity will reside. No design unit may straddle two nodes but must have components on each, which collaborate to accomplish the service.
- Chapter 3 is the Architectural Design. This is the heart of the document. It specifies the design entities that collaborate to perform the functionality of the system. Each of these entities has an Abstract Specification and an Interface that expresses the services that it provides to the rest of the system. In turn each design entity is expanded into a set of lower-level design units that collaborate to perform its services.
- Chapter 4 is the basic Data Structure Design, which for this project is a relational database. While it is separated out here for emphasis, it is really the lowest level of the Architectural Design.
- Chapter 5 is on User Interface Design and discusses the methodology chosen, why it was chosen and why it is expected to be effective.
- Chapter 6 describes the structure of the real-time system.
- Chapter 7 describes the structure of the Help System.
- Chapter 8 exhibits the Use Case Realizations. The implementation of each use case identified in the SRS is shown using the services provided by the design objects.

2. Deployment Diagram

The deployment diagram of this project is as follows(Figure 1).

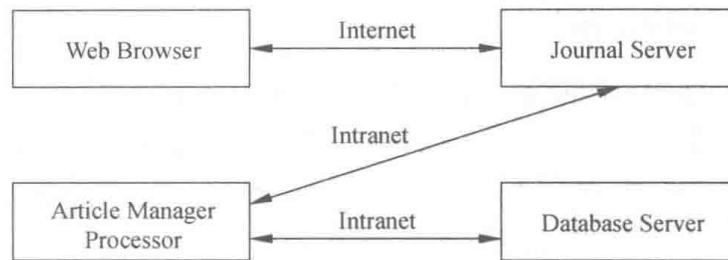


Figure 1 Deployment Diagram

A Reader accesses the On-Line Journal through the Internet using a Web browser (not part of this system although Web pages will run on it). The On-Line Journal resides on a dedicated Journal Server with a permanent Web connection. The Editor manages all of the article preparation work on his/her personal computer (the Article Manager Processor), communicating with the existing Historical Society Database on the Database Server when needed, and uploading completed articles to the Journal Server when they are ready. The Article Manager Processor contains a local file system and the Editor's E-mail system.

3. Architectural Design

The top-level Architectural design of this project is as follows(Figure 2).

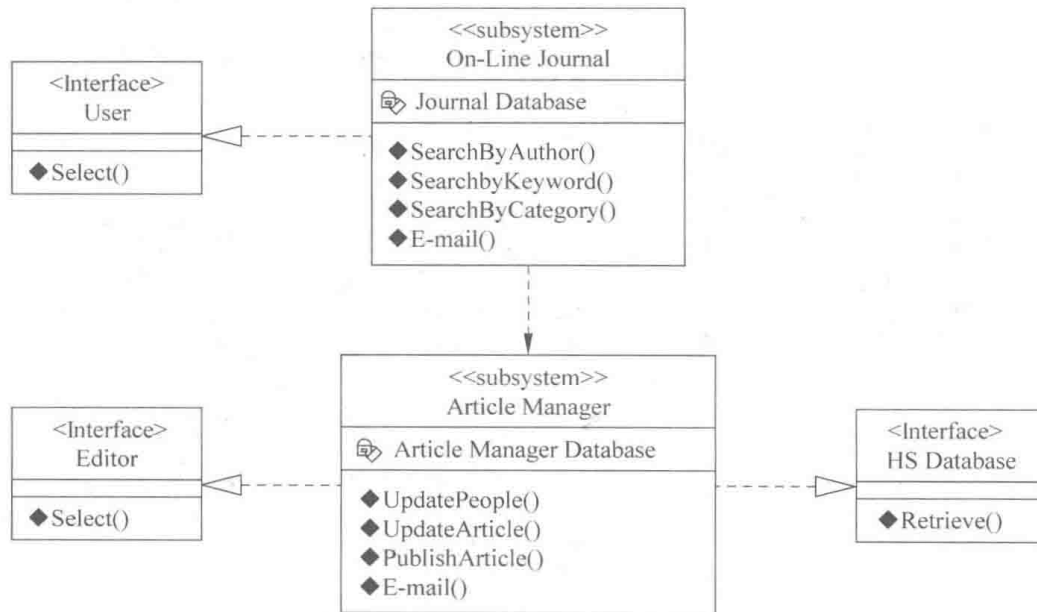


Figure 2 Top-level Architectural Design

User represents any of Author, Reader, or Reviewer. The User Interfaces are covered in Chapter 5.

<< Please note that this top level design starts with the Domain Classes, which are subsystems in the design. *Your* design may not have such natural domain classes. In which case, your architectural design would look more like the design in Figure 3. Do *not* add domain classes unless there are natural ones for your project. >>

3.1 On-Line Journal

Name: On-Line Journal

Type: Subsystem

Node: Journal Server

Description: This is the primary entrance to the system for an Author, Reader or Reviewer. A reader can find and download articles from here. An author or reviewer can access their mail system from here.

Attributes: Journal Database (see Section 4.1)

Resources: Client E-mail system

Operations (detailed below):

SearchByAuthor()

SearchByCategory()

SearchByKeyword()

E-mail()

Unit Design:

The On-Line journal architectural design is as follows(Figure 3).

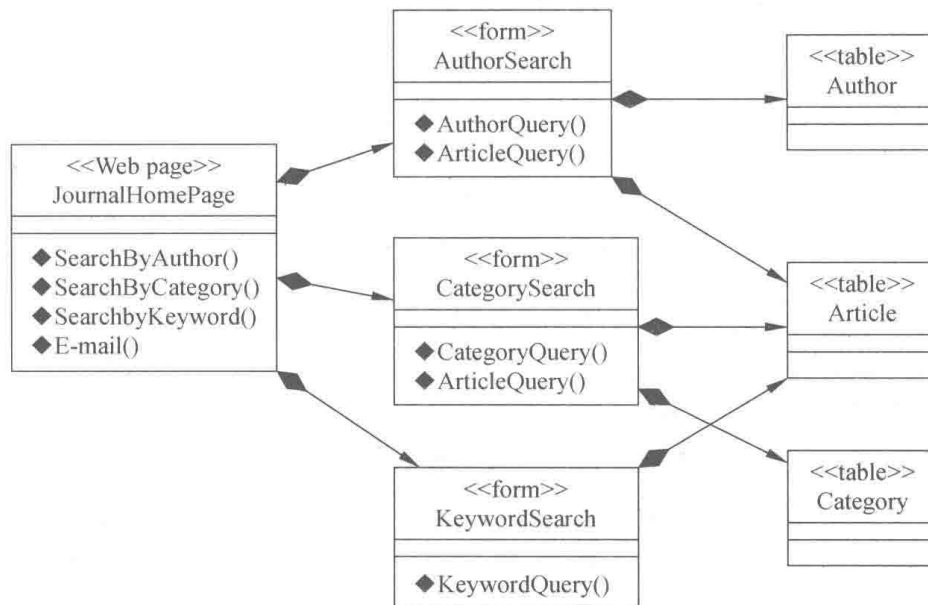


Figure 3 On-Line Journal Architectural Design

3.1.1 On-Line Journal

Name: On-Line Journal

Type: Web Page (This unit has the same name as the subsystem and provides the operations listed there)

Node: Journal Server

Description: This class collects the functionalities of the interface of all users except the Editor. The code would be links on the home page of the Web site and would run on the client's browser.

Attributes: None

Resources: Client's system software

Operations:

Name: SearchByAuthor()

Arguments: None

Returns: File containing article

Description: The Reader will choose an author and an article by that author. An Abstract of the article will appear with the option to download.

Pre-condition: The Reader has accessed the correct Web page.

Post-condition: The article is downloaded to the Reader's PC.

Exceptions: The Reader may abandon the operation without downloading an article.

Flow of Events:

1. The AuthorSearch form is linked to.
2. This form calls AuthorQuery() which presents an alphabetical list of the authors in the database in a pull-down menu.
3. The Reader selects an author from the list.
4. This form calls ArticleQuery() for articles by that author.
5. The Reader selects an article.
6. The Abstract for that Article is displayed on the screen.
7. The Reader selects to download the article or to abandon the action.

Name: SearchByCategory()

Arguments: None

Returns: File containing article

Description: The Reader will choose a category and an article by in that category. An Abstract of the article will appear with the option to download.

Pre-condition: The Reader has accessed the correct Web page.

Post-condition: The article is downloaded to the Reader's PC.

Exceptions: The Reader may abandon the operation without downloading an article.

Flow of Events:

1. The CategorySearch form is linked to.
2. This form calls CategoryQuery() which presents an alphabetical list of the categories in the database in a pull-down menu.
3. The Reader selects a category from the list.
4. This form calls ArticleQuery() for articles in that category.
5. The Reader selects an article.
6. The Abstract for that Article is displayed on the screen.
7. The Reader selects to download the article or to abandon the action.

Name: SearchByKeyword()

Arguments: None

Returns: File containing article

Description: The Reader will enter a keyword (or phrase) and choose an article with that keyword in its Abstract. An Abstract of the article will appear with the option to download.

Pre-condition: The Reader has accessed the correct Web page.

Post-condition: The article is downloaded to the Reader's PC.

Exceptions: The Reader may abandon the operation without downloading an article.

Flow of Events:

1. The KeywordSearch form is linked to.
2. This form presents a text box to enter a keyword.
3. This form calls KeywordQuery() which presents an alphabetical list of the titles of

articles in the database which have that keyword in the article Abstract in a pull-down menu.

4. The Reader selects an article from the list.
5. The Reader selects to download the article or to abandon the action.

Name: E-mail()

Arguments: None

Returns: Access to user's E-mail system

Description: An author or Reviewer selects to pull up the E-mail system on the client PC with the address of the author inserted in the *To*: slot.

Pre-condition: The user has accessed the correct Web page.

Post-condition: The E-mail system with the editor inserted into the *To*: slot is pulled up.

Exceptions: The user may abandon the operation without accessing the E-mail system. If the system is not properly configured, this operation will fail.

Flow of Event:

The system invokes the *mailto*: tag in the HTML code on the Web page.

3.1.2 Author Search Form

Name: AuthorSearch

Type: Form

Node: Journal Server

Description: This form handles a search by author.

Attributes: Author Name, Article Title

Resources: None

Operations:

Name: AuthorQuery()

Arguments: None

Returns: Alphabetical list of authors

Description: This uses the query capability of the database to find an author.

Pre-condition: Control is on the AuthorSearch form.

Post-condition: An author name is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The form upon loading calls a query which returns an alphabetical non-repeating list of authors in the database.
2. The Reader selects an author.

Name: ArticleQuery()

Arguments: Author Name

Returns: Alphabetical list of titles of articles by that author.

Description: This uses the query capability of the database to find the articles.

Pre-condition: Control is on the AuthorSearch form.

Post-condition: An article title is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The form upon loading calls a query which returns an alphabetical list of articles by that author in the database.
2. The Reader selects an article.

3.1.3 Category Search Form

Name: CategorySearch

Type: Form

Node: Journal Server

Description: This form handles a search by category.

Attributes: Category Name, Article Title

Resources: None

Operations:

Name: CategoryQuery()

Arguments: None

Returns: Alphabetical list of categories

Description: This uses the query capability of the database to find a category.

Pre-condition: Control is on the CategorySearch form.

Post-condition: A category name is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The form upon loading calls a query which returns an alphabetical non-repeating list of category names in the database.
2. The Reader selects a category.

Name: ArticleQuery()

Arguments: Category Name

Returns: Alphabetical list of titles of articles in that category

Description: This uses the query capability of the database to find the articles.

Pre-condition: Control is on the CategorySearch form.

Post-condition: An article title is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The form upon loading calls a query which returns an alphabetical list of articles in that category in the database.
2. The Reader selects an article.

3.1.4 Keyword Search Form

Name: KeywordSearch

Type: Form

Node: Journal Server

Description: This form handles a search by keyword.

Attributes: Article Name

Resources: None

Operations:

Name: KeywordQuery()

Arguments: None

Returns: Article Title

Description: This uses the query capability of the database to find articles.

Pre-condition: Control is on the KeywordSearch form.

Post-condition: An article title is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The form presents a text box for the Reader to enter a keyword.
2. The form calls a query which returns an alphabetical list of articles with that keyword in their Abstracts in the database.
3. The Reader selects an article.

3.1.5 Article Table

The Article Table is described in Section 4.1.1

3.1.6 Author Table

The Author Table is described in Section 4.1.2

3.1.7 Category Table

The Category Table is described in Section 4.1.3

3.2 Article Manager

Name: Article Manager

Type: Subsystem

Node: Article Manager Processor

Description: This is the primary entrance to the system for the Editor. All editorial duties can be performed here.

Attributes: Article Manager Database(see section 4.2).

Resources: Access to E-mail, the On-Line Journal database, and the Historical Society Database.

Operations (detailed below):

UpdatePeople()

UpdateArticle()

PublishArticle()

Unit Design:

The article manager architectural design is as follows(Figure 4).

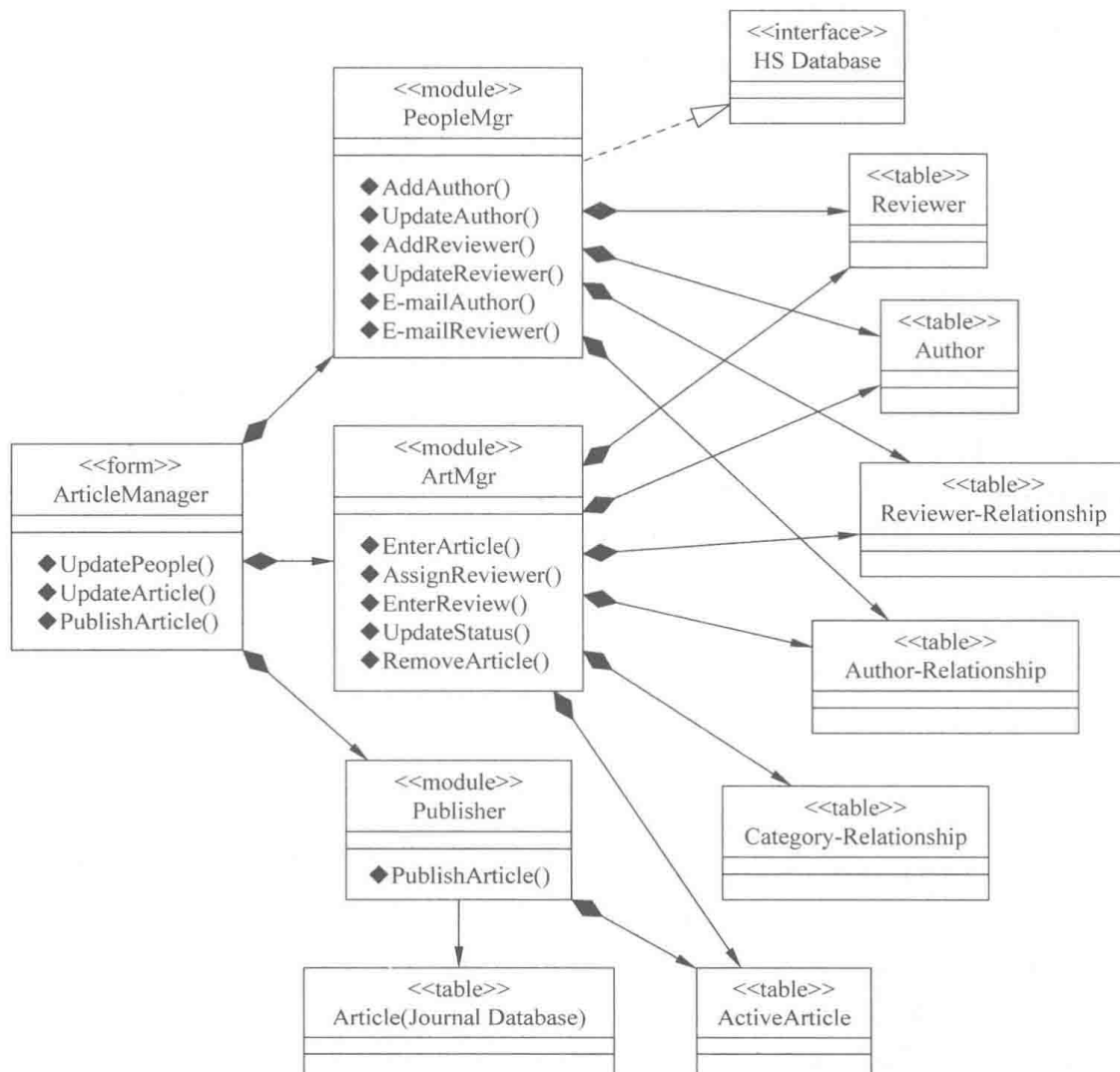


Figure 4 Article Manager Architectural Design

Name: Article Manager

Type: Access Form

Node: Article Manager Processor

Description: This allows the Editor to enter and update Authors, Articles, Reviewers, and to publish Articles.

Attributes: Article Manager Database(see section 4.2).

Resources: Access to E-mail, the On-Line Journal database, and the Historical Society Database.

Operations:

Name: UpdatePeople()

Arguments: None

Returns: None

Description: This is used to enter or update authors and reviewers and to communicate with them.

Pre-condition: Form is active window.

Post-condition: see individual post-conditions.

Exceptions: May be abandoned at any time.

Flow of Event:

The Editor selects the function to be performed.

Name: UpdateArticle()

Arguments: None

Returns: None

Description: This is used to enter or update any information about an article.

Pre-condition: Form is active window.

Post-condition: see individual post-conditions.

Exceptions: May be abandoned at any time.

Flow of Event:

The Editor selects the function to be performed.

Name: PublishArticle()

Arguments: None

Returns: None

Description: This is used to move an article from the Article Manager to the On-Line Journal. It does not remove the article from the Article Manager.

Pre-condition: Form is active window.

Post-condition: Article has been copied to the On-Line Journal.

Exceptions: May be abandoned at any time.

Flow of Event:

The system calls the operation in Publisher.

3.2.1 People Manager Form

Name: PeopleMgr

Type: Access Form

Node: Article Manager Processor

Description: This is used to enter or update authors and reviewers and to communicate with them.

Attributes: Author Name, Reviewer Name

Resources: Access to Historical Society Database

Operations:

Name: AddAuthor()

Arguments: None

Returns: None

Description: This adds a new author to the Article Manager database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a form to fill in Author information.
2. The Editor fills in the form and submits it.
3. The system verifies that required fields have content and enters the new information into the database.

Name: UpdateAuthor()

Arguments: None

Returns: None

Description: This updates author information in the Article Manager database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents an alphabetical drop-down list of Author's names using an Access Query.
2. The Editor selects an Author.
3. The system uses a query to retrieve the information about that author and displays it in a form to update Author information.
4. The Editor updates the form and submits it.
5. The system verifies that required fields have content and enters the updated information into the database.

Name: AddReviewer()

Arguments: None

Returns: None

Description: This adds a new reviewer to the Article Manager database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents an alphabetical list of members' names from the Historical Society Database by executing an Access Query.
2. The Editor selects a name.
3. The system retrieves related information from the HS Database and presents a form to complete with Reviewer information.
4. The Editor fills in the form and submits it.
5. The system verifies that required fields have content and enters the new information into the database.

Name: UpdateReviewer()

Arguments: None

Returns: None

Description: This updates reviewer information in the Article Manager database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents an alphabetical drop-down list of Reviewer's names using an Access Query.
2. The Editor selects a Reviewer.
3. The system uses a query to retrieve the information about that reviewer and displays it in a form to update Reviewer information.
4. The Editor updates the form and submits it.
5. The system verifies that required fields have content and enters the updated information into the database.

Name: E-mailAuthor()

Arguments: None

Returns: Access to user's mail system

Description: The Editor E-mails an Author

Pre-condition: Form is active window and function has been selected.

Post-condition: The E-mail system with the author inserted into the *To:* slot is pulled up.

Exceptions: The user may abandon the operation without accessing the E-mail system. If the system is not properly configured, this operation will fail.

Flow of Events:

The system calls an ActiveX form to access the mail system and fills in the *To*: slot on the E-mail form.

Name: EmailReviewer()

Arguments: None

Returns: Access to user's mail system

Description: The Editor E-mails a Reviewer.

Pre-condition: Form is active window and function has been selected.

Post-condition: The E-mail system with the reviewer inserted into the *To*: slot is pulled up.

Exceptions: The user may abandon the operation without accessing the E-mail system. If the system is not properly configured, this operation will fail.

Flow of Event:

The system calls an ActiveX form to access the mail system and fills in the *To*: slot on the E-mail form.

3.2.2 Article Manager Form

Name: ArtMgr

Type: Access Form

Node: Article Manager Processor

Description: This is used to enter a new article or to update all information about an article.

Attributes: Author Name, Article Name, Reviewer Name

Resources: None

Operations:

Name: EnterArticle()

Arguments: None

Returns: None

Description: This adds a new article to the Article Manager database.

Pre-condition: Form is active window, function has been selected and all authors are already in the database.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a form to fill in Article information.
2. The Editor uses a pull-down menu to assign author(s).
3. The Editor fills in the rest of the form and submits it.
4. The system verifies that required fields have content and enters the new information into the database.

Name: AssignReviewer()

Arguments: None

Returns: None

Description: This adds a reviewer to an article in the Article Manager database.

Pre-condition: Form is active window, function has been selected and the article is already in the database.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a pull-down menu of authors.
2. The Editor selects an author.
3. The system presents a pull-down menu of articles by that author.
4. The Editor selects an article.
5. The system presents a pull-down menu of reviewers.
6. The Editor selects a reviewer.
7. The system enters the new information into the database.

Name: EnterReview()

Arguments: None

Returns: None

Description: This adds a review to an article in the Article Manager database.

Pre-condition: Form is active window, function has been selected and the article with reviewer assigned is already in the database.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a pull-down menu of authors.
2. The Editor selects an author.
3. The system presents a pull-down menu of articles by that author.
4. The Editor selects an article.
5. The system presents a pull-down menu of reviewers for that article.
6. The Editor selects a reviewer and enters the review.
7. The system enters the new information into the database including changing the status of the *review pending*.

Name: UpdateStatus()

Arguments: None

Returns: None

Description: This allows the editor to review and update the status of any article in the Article Manager database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a pull-down menu of authors.
2. The Editor selects an author.
3. The system presents a pull-down menu of articles by that author.
4. The Editor selects an article.
5. The system presents a form with all the information about the article. Appropriate fields will have pull-down menus.
6. The Editor modifies selected fields.
7. The system enters the new information into the database.

Name: RemoveArticle()

Arguments: None

Returns: None

Description: This removes an article in the Article Manager database.

Pre-condition: Form is active window, function has been selected and the article is already in the database.

Post-condition: Database has been modified.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a pull-down menu of authors.
2. The Editor selects an author.
3. The system presents a pull-down menu of articles by that author.
4. The Editor selects an article.
5. The system prompts for confirmation of deletion.
6. The Editor confirms the deletion.
7. The system removes the article entry from the database.

3.2.3 Publisher Form

Name: Publisher

Type: Access Form

Node: Article Manager Processor

Description: This transfers an article from the Article Manager Database to the On-Line Journal Database.

Attributes: None

Resources: None

Operations:

Name: PublishArticle()

Arguments: None

Returns: None

Description: This transfers an article from the Article Manager Database to the On-Line Journal Database.

Pre-condition: Form is active window and function has been selected.

Post-condition: Article is added to On-Line Journal Database.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The system presents a pull-down menu of authors.
2. The Editor selects an author.
3. The system presents a pull-down menu of articles by that author.
4. The Editor selects an article.
5. The system presents a form with the article information.
6. The system accesses the Articles Table of the On-Line Journal Database and downloads a form for a new entry into that table.
7. The system transfers information from the article information form from the Article Manager Database into the article form from the On-Line Journal Database and asks for editing and/or confirmation.
8. The Editor modifies the entry, if desired, and confirms the entry.

3.2.4 Historical Society Interface

Name: Historical Society Database Interface

Type: Interface

Node: Article Manager Processor

Description: Provides membership information about a potential reviewer.

Attributes: None

Resources: Uses Historical Society Database.

Operations:

Name: Retrieve()

Arguments: None

Returns: Selected membership information.

Description: Allows the Editor to search the HS Database for information about a reviewer, who must be a society member.

Pre-condition: Secure access has been established between this system and the HS Database System and an *Add Reviewer Form* is open.

Post-condition: Information is returned.

Exceptions: May be abandoned at any time.

Flow of Events:

1. The Interface accesses the HS Database and retrieves alphabetical list of all members and displays it as a pull-down menu.
2. The Editor chooses an entry from the list.
3. The Interface retrieves selected information from the Database and inserts it into the open Reviewer form.

3.2.5 Author Table

The Author Table is described in Section 4.2.1

3.2.6 Reviewer Table

The Review Table is described in Section 4.2.2

3.2.7 Active Article Table

The Active Article Table is described in Section 4.2.1

3.2.8 Author Relationship Table

The Author Relationship Table is described in Section 4.2.1

3.2.9 Reviewer Relationship Table

The Reviewer Relationship Table is described in Section 4.2.2

3.2.10 Category Relationship Table

The Category Relationship Table is described in Section 4.2.2

3.2.11 Article Table

The Article Table is described in Section 4.1.1

4. Data Structure Design

There are two databases embedded in this product. The first is the On-Line Journal and resides on the Journal Server. The second is the Article Manager Database and resides on the Editor's computer.

4.1 On-Line Journal Database

This database contains three tables. Separating the Author and Article Tables out allows unique storage for the basic article with multiple authors and/or categories supported. The Query feature of the database management system is utilized extensively by the program.

4.1.1 Article Table

Field	Type	Description
ArticleID	Integer	Primary Key
Title	Text	Title of Article
Abstract	Text	Used for Keyword Search, includes Title of Article
Content	Text	Body of Article

4.1.2 Author Table

Field	Type	Description
ArticleID	Integer	Foreign
Author-Last	Text	Last name of an author
Author-First	Text	First name of an author

This table has no primary key. It can be searched for multiple authors for an article.

4.1.3 Category Table

Field	Type	Description
ArticleID	Integer	Foreign
Category	Text	Topic of paper

This table has no primary key. It can be searched for multiple categories for an article.

4.2 Article Manager Database

This database contains six tables. Separating the Relationship Tables out allowed for unique storage of Articles.

4.2.1 Author Table

Field	Type	Description
AuthorID	Integer	Primary Key
Author-Last	Text	Last name of an author
Author-First	Text	First name of an author
E-mail	Text	E-mail address

4.2.2 Reviewer Table

Field	Type	Description
ReviewerID	Integer	Primary Key
Reviewer-Last	Text	Last name of a reviewer
Reviewer-First	Text	First name of a reviewer
E-mail	Text	E-mail Address

4.2.3 Active Article Table

Field	Type	Description
ArticleID	Integer	Primary Key
Title	Text	Title of Article
Abstract	Text	Includes Title of Article
Content	Text	Body of Article

4.2.4 Author Relationship Table

Field	Type	Description
ArticleID	Integer	Foreign
AuthorID	Integer	Foreign

This table has no primary key. It can be searched for multiple authors for an article.

4.2.5 Reviewer Relationship Table

Field	Type	Description
ArticleID	Integer	Foreign
ReviewerID	Integer	Foreign
DateSent	Date	If not yet returned
Review	Text	When returned.
Canceled	Boolean	Review is removed from Reviewer

This table has no primary key. It can be searched to see how many reviews are not yet returned for an article, how many reviews a reviewer has completed, and how many reviews not yet

completed (with the date sent to the reviewer).

4.2.6 Category Relationship Table

Field	Type	Description
ArticleID	Integer	Foreign
Category	Text	Topic of article

This table has no primary key. It can be searched for multiple categories for an article.

5. User Interface Design

5.1 On-Line Journal User Interface

The On-Line User interface will feature the logo of the Historical Society as background. There will be a welcoming message with instructions at the top, three large buttons in the middle, and a small button on the bottom right.

The text of the welcoming message and the instructions has not yet been approved by the Historical Society.

The three buttons will be labeled (from left to right) “Search by Author”, “Search by Category”, and “Search by Keyword”. The small button will be labeled “E-mail Editor”.

When either the Author or Category search is chosen, a pop-up window with a pull-down menu containing an alphabetical, non-repeating list of the appropriate names from the database will appear. When a name is chosen, the pop-up window will be replaced with another pop-up window containing the chosen name at top and all the article titles meeting that criterion.

As each title is selected, the Abstract will appear in the bottom of the form in a scrollable text box. There will be a button to “Select Article”. Once an article is selected, the Web page will invoke a *link* html tag to present the article to open or save.

As authors and categories will be added to the database only when an article referencing them is added, there can be no extra authors or categories. Therefore, no error messages are needed for this part of the page.

The “E-mail Editor” button will invoke a *mailto:* html tag. This tag will invoke the client E-mail server with the *To:* slot of the E-mail header filled in. Any error condition will result in a message from the utility software on the client machine.

5.2 Article Manager User Interface

This will be a standard Access form. It will contain a list of all services grouped by type. The types are “Update People”, “Update Article”, and “Publish Article”.

The details of the various subsequent screens are functionally described in the rest of this document. The Editor has not expressed a desire for any interface beyond a strictly functional one.

As most of the choices are from pull-down menus, there are few opportunities for error messages. The help system can be invoked directly from each page by a small button of the top right corner.

6. Real-Time Design

Real-time considerations are minimal in this project.

The Editor is the sole user of the Article Manager part of the system.

The On-Line Journal is designed for multiple users but the concurrent usage is handled client-side. That is, the Home Page will execute on the client (user's) computer and will make requests of the On-Line Journal server. These requests are handled sequentially by the server with no transient data storage.

7. Help System Design

There is only a minimal Help System for this project.

The On-Line Journal will have a button to E-mail Editor for a user to report problems.

The Article Manager will have a Help choice that will contain the following information:

- a. An overview of the Editor's functionalities extracted from the SRS; and
- b. A list of error messages and details on why they were generated.

In addition, the Help System will contain instructions on how the Editor can bypass the user interface and access the database directly.

8. Use Case Realizations

8.1a Search Article – Author

The sequence diagram of search article by author is as follows(Figure 5).

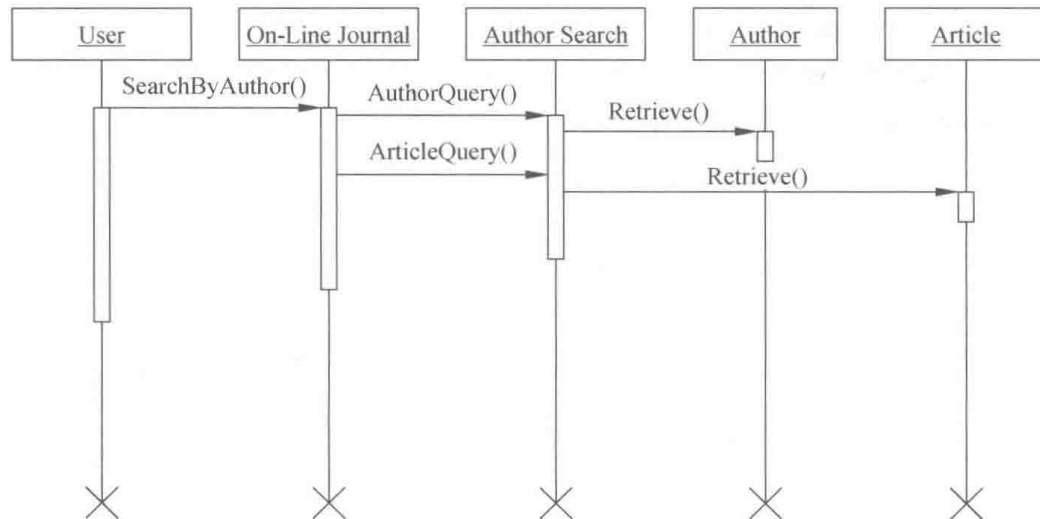


Figure 5 Sequence Diagram of Search Article by Author

Xref: SRS 3.2.1

8.1b Search Article – Category

The sequence diagram of search article by category is as follows(Figure 6).

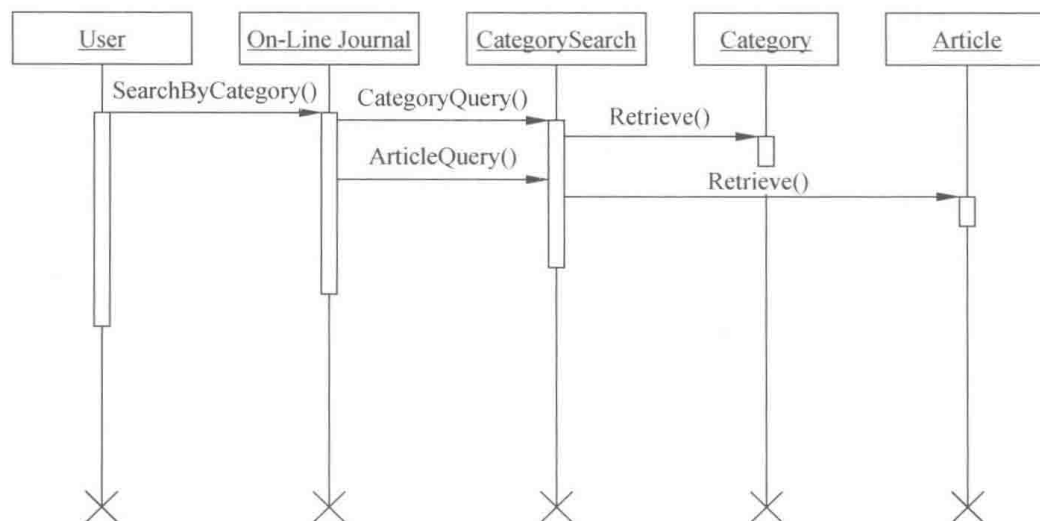


Figure 6 Sequence Diagram of Search Article by Category

Xref: SRS 3.2.1

8.1c Search Article – Keyword

The sequence diagram of search article by keyword is as follows(Figure 7).

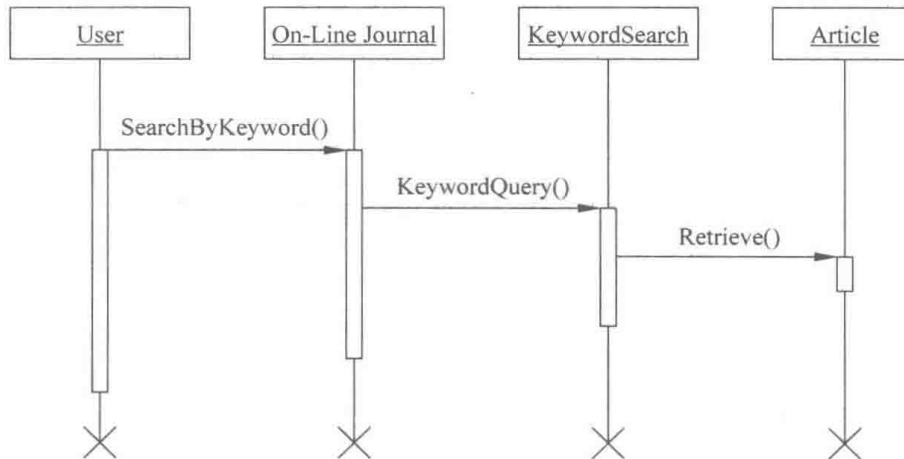


Figure 7 Sequence Diagram of Search Article by Keyword

Xref: SRS 3.2.1

8.2 Communicate

The sequence diagram of communicate is as followa(Figure 8).

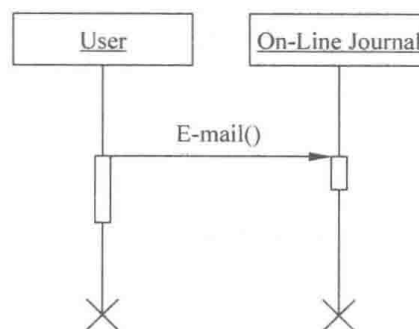


Figure 8 Sequence Diagram of Communicate

Xref: SRS 3.2.2

8.3 Add Author

The sequence diagram of add author is as follows(Figure 9).

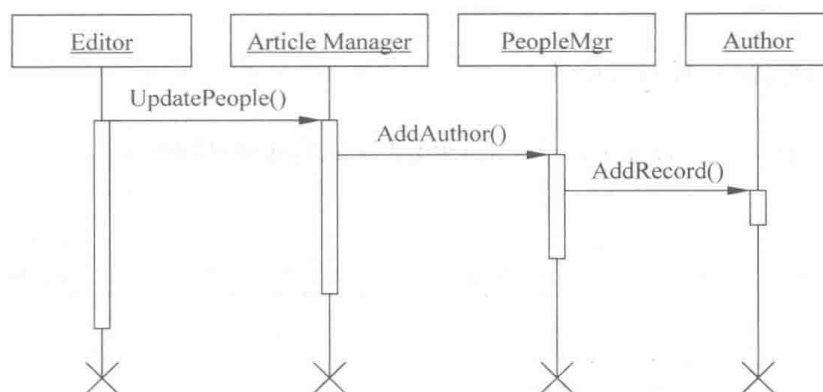


Figure 9 Sequence Diagram of Add Author

Xref: SRS 3.2.3

8.4 Add Reviewer

The sequence diagram of add reviewer is as follows(Figure 10).

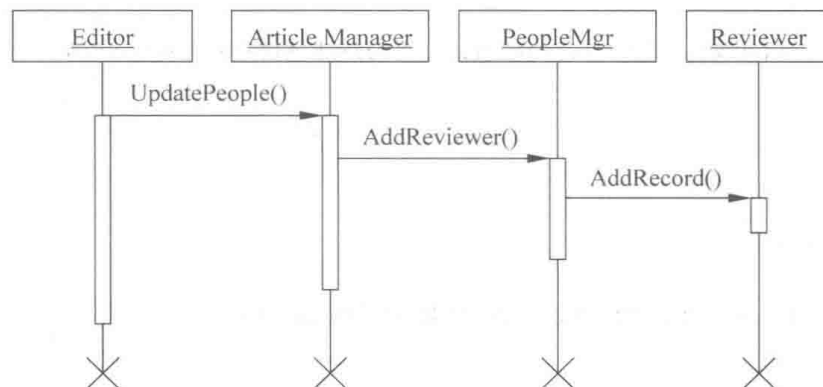


Figure 10 Sequence Diagram of Add Reviewer

Xref: SRS 3.2.4

8.5a Update Person – Author

The sequence diagram of update person by author is as follows(Figure 11).

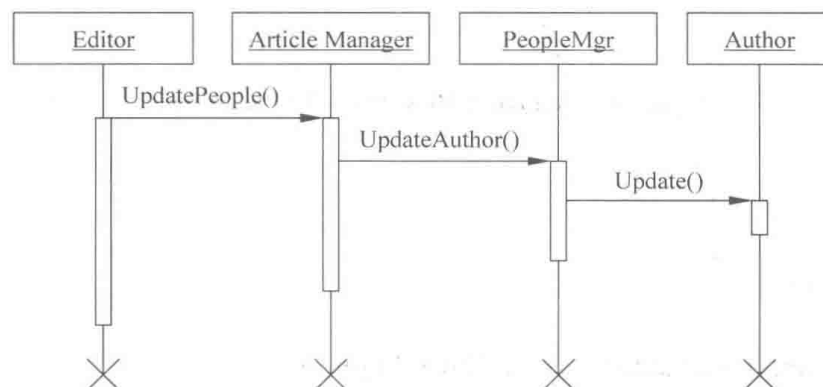


Figure 11 Sequence Diagram of Update Person by Author

Xref: SRS 3.2.5

8.5b Update Person – Reviewer

The sequence diagram of update person by reviewer is as follows(Figure 12).

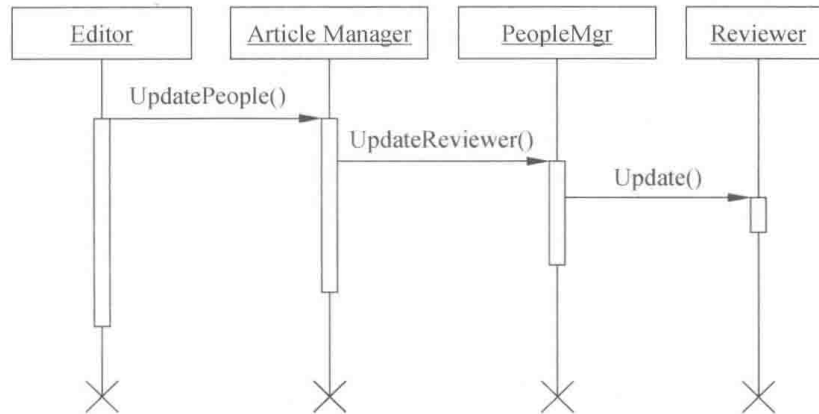


Figure 12 Sequence Diagram of Update Person by Reviewer

Xref: SRS 3.2.5

8.6a Update Article Status – Delete Reviewer

The sequence diagram of update article status by delete reviewer is as follows(Figure 13).

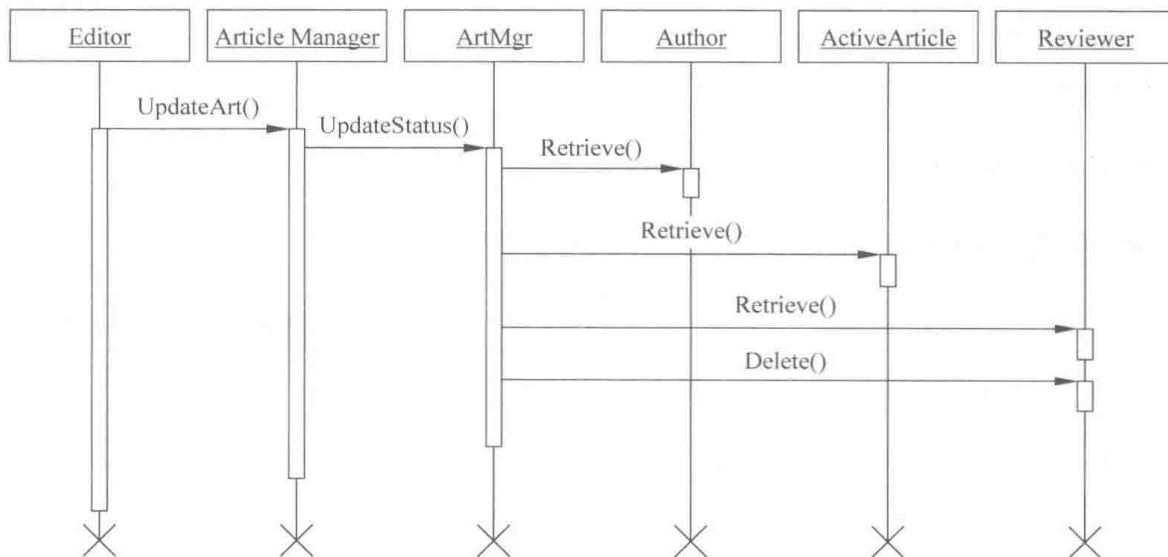


Figure 13 Sequence Diagram of Update Article Status by Delete Reviewer

Xref: SRS 3.2.6

<<There actually should be more of these realizations for Use Case 3.2.6>>

8.6b Update Article Status – Add/Delete Category

The sequence diagram of update article status by add /delete category is as follows(Figure 14).

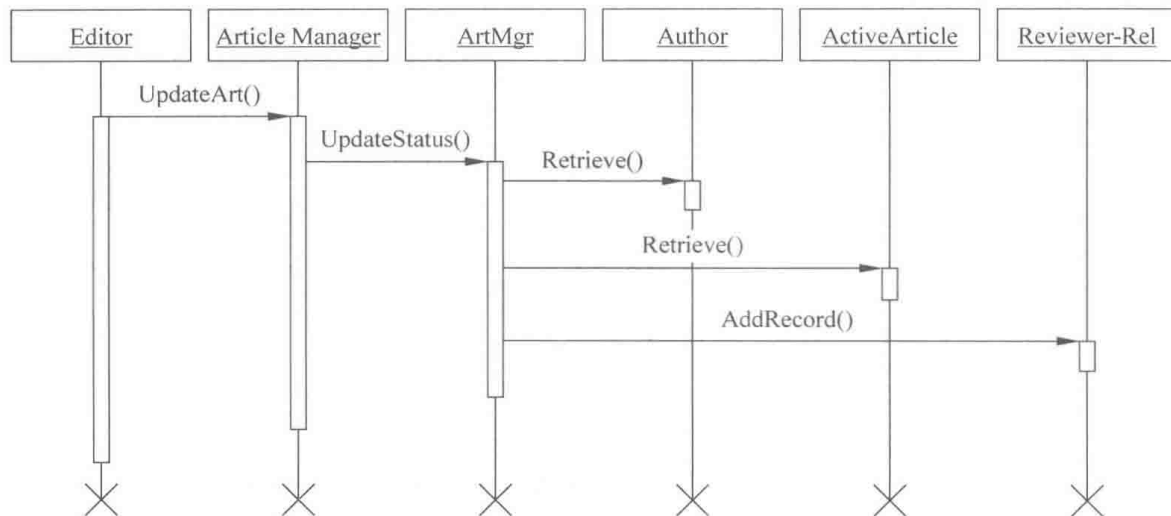


Figure 14 Sequence Diagram of Update Article Status by Add /Delete Category

Xref: SRS 3.2.6

8.7 Enter Communication

In the above, when the article is retrieved, modify its information and then update the record in the table with this modified information.

Xref: SRS 3.2.7

8.8 Assign Reviewer

The sequence diagram of assign reviewer is as follows(Figure 15).

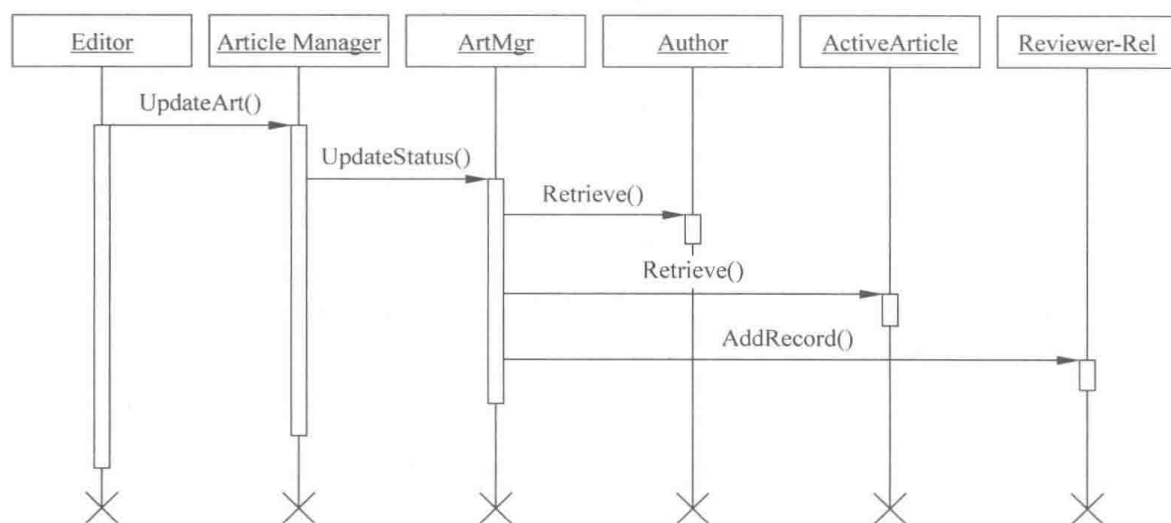


Figure 15 Sequence Diagram of Assign Reviewer

8.9 Check Status

Use 8.6 above but make no changes.

Xref: SRS 3.2.9

8.10 Send Communication

The sequence diagram of send communication is as follows(Figure 16).

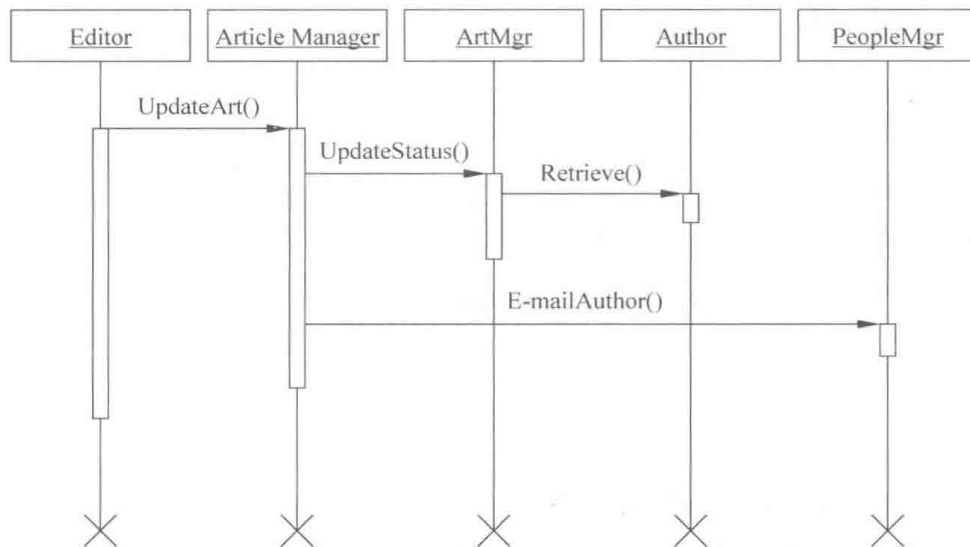


Figure 16 Sequence Diagram of Send Communication

Xref: SRS 3.2.10

8.11 Publish Article

The sequence diagram of publish article is as follows(Figure 17).

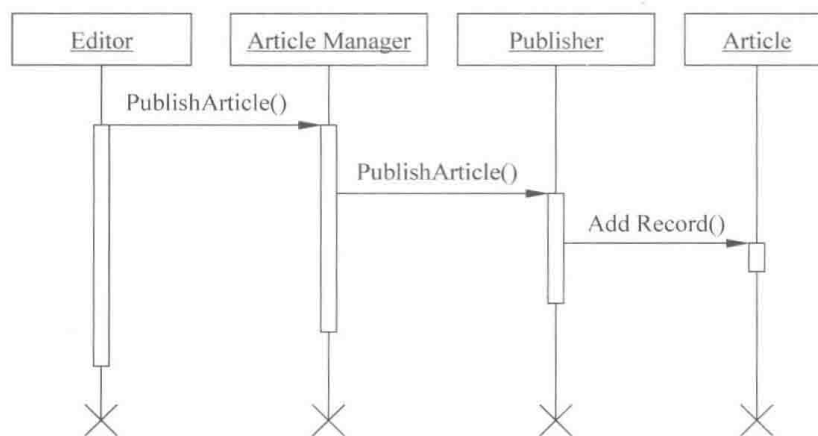


Figure 17 Sequence Diagram of Publish Article

Xref: SRS 3.2.11

8.12 Remove Article

The sequence diagram of remove article is as follows(Figure 18).

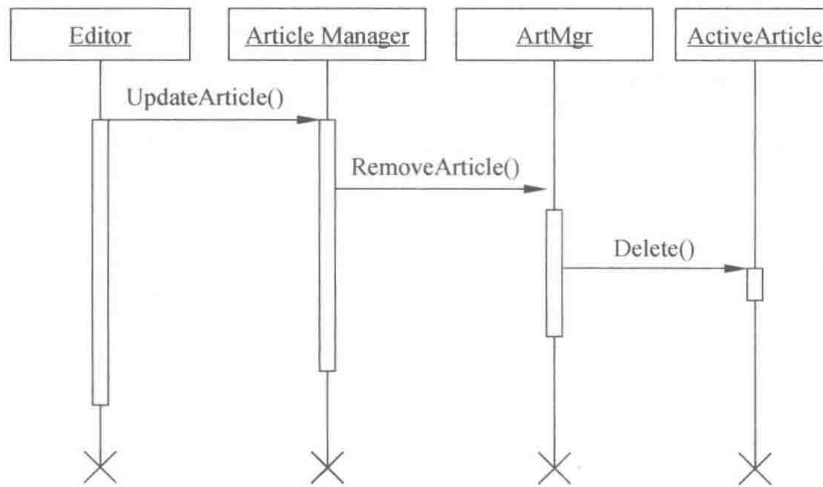


Figure 18 Sequence Diagram of Remove Article

Xref: SRS 3.2.12

Exercises

1. What does this document mainly describe?
2. How many functions does Web Publishing System provide?
3. What is Article Manager Database mainly used for?

Implementing the System

系统实现

Part 1 Listening & Speaking

➤ Dialogue: Creating High-Quality Code

Kevin: Before coding, I think it is necessary to **conform to** some principles during coding. **With the help of** these guidelines, we can be more reliable to create high-quality code and use the advantages of C# sufficiently, a popular object-oriented language which we have decided to use for coding.

Jason: For Object-Oriented Programming, ^[1] programmers should **think about** programming in terms of classes, I think. A class is a collection of data and routines that share a cohesive, well-defined responsibility, and might also be a collection of routines that provides a cohesive set of services even if no common data is involved. A key to being an effective programmer is maximizing the portion of a program that you can safely ignore while working on any one section of code. Classes are the primary tool for accomplishing that objective.

Sharon: I absolutely agree with you. **Furthermore**, I think that the first and probably the most important step in creating a high-quality class is creating a good interface. This consists of creating a good abstraction for the interface to represent and ensure that the details remain hidden behind the abstraction. Particularly, in the analysis and design phase of our system, we have already identified all the classes, and defined the attributes and operation relevant for each of them. Now we should review and modify the classes with these criteria, which should be held constant when coding all of them.

Kevin: Yes. **In addition**, ^[2] another important factor is high-quality routines within the classes. The routine makes programs easier to read and

- [1] Replace with:
1. Considering Object-Oriented Programming
 2. **With an eye to** Object-Oriented Programming
 3. **With a view to** Object-Oriented Programming

- [2] Replace with:
1. Besides
 2. Furthermore
 3. Moreover

understand than any other feature of any programming language, and is also the greatest technique ever invented for saving space and improving performance, I think. Imagine how much larger our code would be if we had to repeat the code for every call to a routine instead of **branching** to the routine. Imagine how harder it would be to make performance improvements in the same code used in a dozen places instead of making them all in one routine. Once we've identified the class's major routines in the first step, we must construct each specific routine. Construction of each routine typically **unearths** the need for additional routines, both **minor** and **major**, and issues arising from creating those additional routines often **ripple** back to the overall class design.

Jason: So constructing classes and constructing routines tends to be an iterative process.

Kevin: That's right. In every iterative process, we will test each routine as it's created. After the class as a whole becomes operational, we will review and test the class as a whole for any issues that can't be tested at the individual-routine level. And at each step, we should check our work and encourage others to check it too. In that way, we'll catch mistakes at the least expensive level, when we've invested the least amount of effort.

Sharon: Besides, the Pseudocode Programming Process is a useful tool for detailed design and makes coding easy. Pseudocode can be translated directly into **comments**, ensuring that the comments are accurate and useful.

Words

routine

[ru:'ti:n] n. 程序

furthermore

[fə:'ðə'mɔ:] adv. 此外, 而且

branch

[brɑ:ntʃ] v. 分支

unearth

['ʌn'ə:θ] v. 发现, 揭露

minor

['maɪnə] adj. 较小的, 次要的

major

['meɪdʒə] adj. 较大的, 较重要的

ripple

['rɪpl] v. 起伏波动

pseudocode

['psju:dəukəʊd] n. 伪代码

comment

['kɒment] n. 注释, 注解

moreover

[mɔ:'rəʊvə] adv. 而且, 此外

Phrases

conform to

遵守

with the help of

借助

think about

考虑, 思考

in addition

此外

with an eye to

着眼于

with a view to

为了, 目的在于

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ Listening Comprehension: Writing the Code

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which point of view is not correct on the purpose of matching implementation to design?
(A) Easy to implement all the structures and relationships that are described with the charts and tables in the detailed design document as code directly
(B) Easy to trace the algorithms, functions, interfaces, and data structures from design to code and back again
(C) Easy to integrate all of the programmers' code into the whole system
(D) Easy for testing, maintenance, and configuration management over time
- Which advantage of the design characteristics is not referred in this article?
(A) Low coupling
(B) Well-defined interfaces
(C) High cohesion
(D) Hierarchical building
- Which technique is not mentioned in this article on coding with your team in mind?
(A) Standards
(B) Documents
(C) Common design techniques and strategies
(D) Information hiding

Words

daunt

[dɔːnt] v. 沮丧, 使气馁

correspondence

[kɒrɪs'pɒndəns] n. 通信

modularity

[ˌmɒdju'lærɪti] n. 模块性

coupling

['kʌplɪŋ] n. 耦合

cohesion

[kəu'hiːʒən] n. 内聚

algorithm

['ælgərɪðəm] n. 算法

function

['fʌŋkʃən] n. 函数

invoke

[ɪn'vəʊk] v. 调用

Phrases

take advantage of

利用

keep in mind

记住

➤ Dictation: Concentrate on the Vital Few, Not the Trivial Many

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

The Pareto principle, also known as the 80-20 rule, states that, for many events, 80 percent of the 1 comes from 20 percent of the 2. In other words, you can get 80 percent of the result with 20 percent of the effort. It is a statistical method to identify the 3 of agents that exert the 4 effect.

Business management thinker Joseph M. Juran, father of Pareto principle in 5 management, suggested the principle and 6 it after Italian 7 Vilfredo Pareto, who observed that 80 percent of income in Italy went to 20 percent of the 8. It is a common **rule of thumb** which **applies to** a lot of areas. e.g., “80 percent of your sales comes from 20 percent of your clients” in business.

9, there are many software phenomena 10 Pareto distribution. For example, 20 percent modules consume 80 percent resources, 20 percent modules 11 80 percent errors, 20 percent errors consume 80 percent repair costs, 20 percent modules consume 80 percent 12 time, and 20 percent tools 13 80 percent usage, and so on.

One of definite applications of the software phenomena above is 14 optimization. Donald Knuth **profiled** his line-count program and found that it was spending half its execution time in two 15. He changed a few lines of code and 16 the speed of the **profiler** in less than an hour. It is advised that you should measure the code to find the 17 and then put your 18 into optimizing the few 19 that are used the most.

The Pareto Principle can and should be used by every intelligent person in their daily life. It can 20 your **profitability** and effectiveness.

Words

vital

['vaɪtəl] adj. 生死攸关的, 至关重要的

trivial

['trɪviəl] adj. 琐细的, 价值不高的, 微不足道的

agent

['eɪdʒənt] n. 使然力 (指引起一定作用的人或其他因素)

exert

[ɪg'zɜ:t] v. 尽 (力), 施加 (压力等)

Words**thinker**

['θɪŋkə] n. 思想者

profile

['prəʊfaɪl] v. 用数据图表（或曲线图）表示

profiler

['prəʊfaɪlə] n. 曲线图，概览图

profitability

[ˌprɒfɪtə'bɪlɪtɪ] n. 收益性，利益率

Phrases

rule of thumb

单凭经验的方法

apply to

应用

Part 2 Reading & Translating**➤ Section A: Computer Programming**

A program is a list of instructions that the computer must follow to process data into information. The instructions consist of **statements** used in a programming language, such as Basic. Examples are programs that do word processing, desktop publishing, or payroll processing. Programming is a five-step process for creating that lists of instructions as follows:

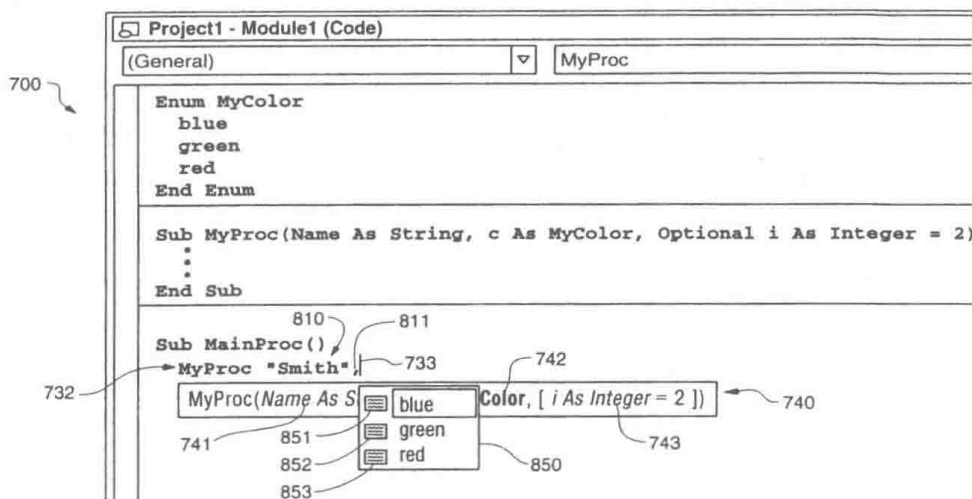
1. The problem clarification (definition) step consists of six mini-steps—clarifying program objectives and users, outputs, inputs, and processing tasks; studying the feasibility of the program; and documenting the analysis.
2. To design the solution, one first needs to create an algorithm. An algorithm is a formula or sets of steps for solving a particular problem. In the program design step, the software is designed in two mini-steps. First, the program logic is determined through a top-down approach and modularization, using a hierarchy chart (or structure chart). Then it is designed **in detail**, either in **narrative** form, using pseudocode, or graphically, using **flowcharts**. Pseudocode is a method of designing a program using normal human-language statements to describe the logic and the processing flow. Most programmers use a design approach called structured programming. Structured programming takes a top-down approach that breaks programs into modular forms. In structured program design, three control structures are used to form the logic of a program: sequence, selection, and iteration (or loop).
3. Once the design has been developed, the actual writing of the program begins. Writing the program is called coding. **Coding** is what many people think of when they think of programming, although it is only one of the five steps. Coding consists of translating the logic requirements from pseudocode or flowcharts into a programming language — the **letters**, numbers, and symbols that

make up the program. A programming language is a set of rules that tells the computer what operations to do. Examples of well-known programming languages are C, C++, and Java. These are called high-level languages.

4. Program testing involves running various tests and then running real-world data to make sure the program works. Two principal activities are **desk-checking** and debugging. These steps are called **alpha testing**. Desk-checking is simply **reading through**, or checking, the program to make sure that it's **free of** errors and that the logic works. Once the program has been desk-checked, further errors, or "bugs" will doubtless **surface**. To debug means to detect, locate, and remove all errors in a computer program. Mistakes may be syntax errors or logical errors. Syntax errors are caused by **typographical** errors and incorrect use of the programming language. Logic errors are caused by incorrect use of control structures. After desk-checking and debugging, the program may run fine—in the laboratory. However, it needs to be tested with real data; this is called **beta testing**.
5. Writing the program documentation is the fifth step in programming. The resulting documentation consists of written descriptions of what a program is and how to use it. Documentation is not just an **end-stage** process of programming. It has been (or should have been) going on **throughout** all programming steps. Documentation consists of user documentation, operator documentation, programmer documentation, and it can be used to maintain the program.

Programming languages have five generations. The births of the generations are as follows:

- (1) First generation, 1945—machine language
- (2) Second generation, mid-1950s—assembly language
- (3) Third generation, mid-1950s to early 1960s—high-level languages (procedural languages and object-oriented); for example, Fortran, COBOL, BASIC, C, and C++
- (4) Fourth generation, early 1970s—very high-high-level languages (problem-oriented languages); for example, SQL, Intellect, NOMAD, FOCUS
- (5) Fifth generation, early 1980s—natural languages



A **compiler** is a language-translator program that converts the entire program of high-level language into machine language before the computer executes the program. Examples of procedural languages using compilers are COBOL, FORTRAN, Pascal, and C. An **interpreter** is a language-translator program that converts each procedural language statement into machine language and executes it immediately, statement by statement. BASIC is a procedural language using an interpreter. When a compiler is used, it requires two steps (the source code and the object code) before the program can be executed. The interpreter, on the other hand, requires only one step. The advantage of a compiler language is that, once you have obtained the object code, the program executes faster. The advantage of an interpreter language, on the other hand, is that programs are easier to develop. Some language translators—such as those with C++ and Java—can both compile and interpret.

In object-oriented programming, data and the instructions for processing that data are combined into a self-sufficient “**object**” that can be used in other programs. An object is a self-contained module consisting of **preassembled** programming code. Send the “message” when an object’s data is to be processed. A message is an alert sent to the object when an operation involving that object needs to be performed. The message needs only to identify the operation. How it is actually to be performed is embedded within the processing instructions that are part of the object. These processing instructions within the object are called the methods.

Object-oriented programming has three important concepts, which **go under** the **jaw-breaking** names of **encapsulation**, inheritance, and polymorphism. [1] Encapsulation means an object contains (encapsulates) both data and relevant processing instructions. Once an object has been created, it can be reused in other programs. An object’s uses can also be extended through concepts of class and inheritance. All objects that are **derived from** or **related to** one another are said to form a class. Each class contains specific instructions (methods) that are unique to that group. Inheritance is the method of **passing down** traits of an object from classes to **subclasses** in the hierarchy. Thus, new objects can be created by inheriting traits from existing classes. Polymorphism means that a message (generalized request) produces different results based on the object that it is sent to.

Examples of object-oriented programming languages are C++ & Java. C++—the plus sign stands for “more than C”—**combines** the traditional C programming language **with** object-oriented capability. C++ was created by Bjarne Stroustrup. With C++, programmers can write standard code in C without the object-oriented features, use object-oriented features, or do a mixture of both. A high-level programming language developed by Sun Microsystems in 1995, Java is used to write **compact** programs that can be **downloaded** over the Internet and immediately executed on many kinds of computers. Java is similar to C++ but is simplified to eliminate language features that cause common programming errors.

A **markup language** is a kind of coding, or “tags,” inserted into text that **embeds** details about the structure and appearance of the text. HTML (HyperText Markup Language) is a markup language that lets people create on-screen documents for the Internet that can easily be linked by words and pictures to other documents. VRML (Virtual Reality Modeling [or Markup] Language) is a type of programming

language used to create three-dimensional Web pages including interactive **animation**. XML (Extensible Markup Language) is a **metalanguage** (a language used to define another language) written in SGML that allows one to facilitate the easy interchange of documents on the Internet.

A **script** is a short list of self-executing commands embedded in a Web page that perform a specific function or routine. JavaScript is a popular object-oriented scripting language that is widely supported in Web **browsers**. It adds interactive functions to HTML pages, which are otherwise static. **ActiveX** is a set of control, or reusable components, that enable programs or content of almost any type to be embedded within a Web page. **Perl** (Practical Extraction and Report Language) is a general-purpose programming language developed for text manipulation and now used for Web development, network programming, system administration, GUI development, and other tasks.

Words

statement

['steitmənt] n. 语句

narrative

['nærətɪv] adj. 叙述的

flowchart

['fləʊtʃɑ:t] n. 流程图

code

[kəʊd] v. 编码

letter

['letə] n. 字母

bug

[bʌg] n. 程序缺陷

surface

['sə:fɪs] v. 出现, 显露

typographical

[.taɪpə'græfɪkəl] adj. 印刷上的, 排字上的

throughout

[θru:'aʊt] prep. 遍及, 贯穿

generation

[dʒenə'reɪʃən] n. 一代, 阶段

assembly

[ə'sembli] n. 汇编

compiler

[kəm'paɪlə] n. 编译器

interpreter

[ɪn'tə:prɪtə] n. 解释程序, 解释器

object

['ɒbdʒɪkt] n. 对象

preassemble

['pri:ə'sembl] v. 预装(配), 预先安装

encapsulation

[ɪn,kæpsju'leɪʃən] n. 封装

trait

[treɪt] n. 特性

subclass

['sʌbklɑ:s] n. 子类

compact

[kəm'pækt] adj. 简明的, 紧凑的

download

['daʊnləʊd] v. 下载

embed

[ɪm'bed] v. 嵌入

animation

[æni'meɪʃən] n. 动画

metalanguage

['metə.læŋgwidʒ] n. 元语言

script

[skript] n. 脚本

browser

[braʊzə] n. 浏览器

Phrases

in detail	详细地
desk-checking	部件检验
alpha testing	α 测试
reading through	通读
free of	没有……的
beta testing	β 测试
end-stage	末期的
go under	归为（某一类事物）中
jaw-breaking	难发音的
be derived from	源于，得自
relate to	与……有关
pass down	把……传下去
combine with	使……与……联合
markup language	标识语言

Abbreviations

HTML	HyperText Markup Language	超文本标记语言
VRML	Virtual Reality Modeling [or Markup] Language	虚拟现实建模（标记）语言
XML	eXtensible Markup Language	可扩展标记语言
SGML	Standard for General Markup Language	通用标记语言标准
ActiveX		微软公司倡导的网络化多媒体对象技术
Perl	Practical Extraction and Report Language	实用报表提取语言
GUI	Graphical User Interface	图形用户界面

Complex Sentences

Original: Object-oriented programming has three important concepts, which go under the jaw-breaking names of encapsulation, inheritance, and polymorphism.

Translation: 面向对象程序设计有三个重要的概念，分别是封装、继承和多态。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

___ 1. A program consists of instructions which are composed of statements used in a

programming language.

- ___ 2. "Programming" is just "coding" that many people usually think of.
- ___ 3. Sequence, selection, and iteration are the three control structures which are used to form the logic of a program in structured program design.
- ___ 4. Program testing involves two steps: desk-checking and debugging.
- ___ 5. A message is a set of processing instructions to perform the operations of the objects involved.

II. Choose the best answer to each of the following questions.

1. Which gradation is right about the steps in the programming process according to this article?
 - (A) Problem clarification, program design, program coding, alpha testing, program documentation.
 - (B) Problem clarification, algorithm creating, program documentation, program coding, program testing, program documentation.
 - (C) Problem definition, pseudocode, program coding, beta testing, program documentation.
 - (D) Problem definition, program design, program coding, program testing, program documentation.
2. Which of the following is right about the program languages according to this article?
 - (A) With C++, programmers must use object-oriented features which are different from the standard code in C.
 - (B) All the languages need to be converted by both language-translator programs into machine language to execute.
 - (C) Compared to C++, Java is merely simplified to eliminate some complex language features.
 - (D) Polymorphism allows a message to produce different results based on the object that it is sent to.
3. Which of the following is wrong about the differences between compiler and interpreter?
 - (A) A compiler converts the entire program of high-level language into machine language before the computer executes the program.
 - (B) An interpreter converts each procedural language statement into machine language and executes it immediately.
 - (C) When a compiler requires two steps before the program can be executed, the interpreter requires only one step, so the interpreter enables the program to execute faster.
 - (D) An interpreter language enables the program to be developed more easily.

III. Translation

Pseudocode is a method of designing a program using normal human-language statements to describe the logic and the processing flow.

➤ Section B: Location-based Service 基于位置的服务

Location-based software is able to access your current location and use it to show you the closest shops, restaurants, and theaters, as well as information about each location. Want to find the nearest ATM? Need **turn-by-turn** directions to the airport? Want to **hook up with** friends who are nearby? Would you like some user reviews about the Mexican restaurant you just **spotted**? There are location-based apps designed to answer those questions and more. Location-based software is available for desktop, laptop, and handheld computers. Mobile devices can **pinpoint** your location using the **built-in GPS**(Global Positioning System) or by **triangulating** your distance from nearby **cell towers**. Desktop and laptop computers can determine your location based on your Internet service provider and nearby private computer networks.



When devices record your location, there is the possibility of abuse. Stay alert for devices and software applications that track your location. In some cases, you can turn tracking off temporarily or permanently. In other cases, tracking may not be under your control; you'll have to decide whether the service you receive from the device or software is worth **relinquishing** your privacy.

Location-based services (LBS) are a general class of computer program-level services used to include specific controls for location and time data as control features in computer programs. Such LBS is an information service and has a number of uses in social networking today as an entertainment service, which is accessible with mobile devices through the mobile network and which uses information on the geographical position of the mobile device. [1] This has become more and more important with the expansion of the smartphone and tablet markets as well.

LBS are used in a variety of contexts, such as health, indoor object search, entertainment, work, personal life, etc.

LBS include services to identify a location of a person or object, such as discovering the nearest banking cash machine (also known as **ATM**) or the **whereabouts** of a friend or employee. LBS include parcel tracking and vehicle tracking services. LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location.

They include personalized weather services and even location-based games. They are an example of telecommunication convergence.

Location-based services like Foursquare, Gowalla, Where, and Facebook Places use the geo-location functionality of a mobile phone or smart phone to provide people with information and entertainment.

These services allow your customers to “check-in” wherever they go—like coffee shops, events, restaurants, hotels, airports, or even private residences—**in exchange for** both tangible and intangible **rewards**.

OK, it might seem crazy at first that people would “check-in”, but they do (and in very large numbers)! The truth is that people check-in for a number of reasons... to tell their friends where they are, to keep a record of what they do, to earn gaming rewards like **badges** and **mayorships**, and to unlock **specials** from participating businesses and venues that reward check-ins. [2]

As a business owner, you can use these features to offer specials, discounts, and prizes for your loyal customers.

LBS is so wonderful!

Words

spot

[spɒt] v. 发现, 认出

pinpoint

['pinpoint] v. 精确地找到, 准确描述

triangulate

[traɪ'æŋɡjuleɪt] v. 对……作三角测量

relinquish

[rɪ'liŋkwɪʃ] v. 放弃, 放手, 让渡

whereabout

['werəbaʊt] n. 所在之处, 行踪, 下落

reward

[rɪ'wɔ:d] n. 报答, 报酬

badge

[bæɡɜ:] n. 徽章, 证章, 标记

mayorship

['meəʃɪp] n. 市长的职位

special

['speʃ(ə)l] n. 特价商品

Phrases

turn-by-turn

路线规划, 建议路线, 分路段显示

hook up with

沟通, 交流

cell towers

基站, 基地台

in exchange for

作为……的交换, 用……交换

Abbreviations

LBS	Location-Based Services	基于位置的服务
ATM	Automatic Teller Machine	自动出纳机

Complex Sentences

[1] **Original:** Such LBS is an information service and has a number of uses in social networking today as an entertainment service, which is accessible with mobile devices through the mobile network and which uses information on the geographical position of the mobile device.

Translation: 这种基于位置的服务是一种信息服务并且在当前的社交网络中以娱乐服务的形式拥有一定数量的用户。这些用户可以通过移动设备的移动网络或地理位置信息来使用这种服务。

[2] **Original:** The truth is that people check-in for a number of reasons... to tell their friends where they are, to keep a record of what they do, to earn gaming rewards like badges and mayorships, and to unlock specials from participating businesses and venues that reward check-ins.

Translation: 事实上,人们会“签到”源自多方面原因,包括告诉他们的朋友自己的位置、记录自己的行踪、赢取诸如徽章和市长职位(或头衔)的游戏奖励,以及从参加奖励“签到”的商业活动和场所中来解锁的特价商品。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Location-based software can be only available for handheld computers.
- ___ 2. When devices record your location, there is no possibility of abuse.
- ___ 3. Foursquare is not a location-based service.
- ___ 4. LBS can probably help you to find the nearest shopping center.
- ___ 5. Location-based software is able to access your current location and use it to show you the fastest schools, restaurants, and theaters, as well as information about each location.

II. Choose the best answer to each of the following questions.

1. Which of the following is right about the LBS according to this article?
 - (A) LBS are something similar to Microsoft Excel.
 - (B) LBS include services to identify a location of a person or object, such as discovering the nearest shopping mall or the whereabouts of a friend or employee.
 - (C) LBS are just concepts, no applications now.
 - (D) It is not possible to abuse LBS.

2. Which of the following is wrong about LBS according to this article?
 - (A) LBS are used in a variety of contexts, such as health, indoor object search, entertainment, work, personal life, etc.
 - (B) LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location.
 - (C) Location-based software is available for desktop, laptop, and handheld computers.
 - (D) LBS have no privacy problems.
3. Which of the following is not LBS related?
 - (A) Microsoft Word
 - (B) Facebook Places
 - (C) Foursquare
 - (D) Gowalla

III. Translation

1. Location-based services (LBS) are a general class of computer program-level services used to include specific controls for location and time data as control features in computer programs.
2. LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location.

Part 3 Simulated Writing: Progress Report

Functions and Contents of Progress Reports

Progress reports are common in engineering. Once you have written a successful proposal and have secured the resources to do a project, you are expected to update the client on the progress of that project. This updating is usually handled by progress reports. They might be one-page memos or long, letters, short reports, formal reports, or presentations. Such a report is aimed at whoever assigned the project. Your goals are to give a fair assessment of what you have accomplished on the project and of what you expect to do in the future, and to enable the manager or sponsor of a project to make informed decisions about the future of the project. Yet, any project of size or significance is bound to encounter snags: additional requirements, miscommunications, problems, delays, or unexpected expenses. A progress report must account for those snags. You should discuss any problems you are having and what further help you require.

What information expected in a progress report depends on the situation, but most progress reports have the following similarities in content:

- 1) Background on the project itself
- 2) Discussion of achievements since last reporting
- 3) Discussion of problems that have arisen

- 4) Discussion of work that lies ahead
- 5) Assessment of whether you will meet the objectives in the proposed schedule and budget

Organizational Patterns for Progress Reports

The structure of the progress report is determined by the original proposal for the project: make use of original milestones or the timeline. With this in mind, a more comprehensive list of components will give you a clearer structure as follows.

- 1) Introduction
- 2) Project Description
- 3) Work Completed
- 4) Problems Arisen
- 5) Changes in Requirements
- 6) Work Scheduled
- 7) Overall Assessment of the Project

1. Introduction

As always, first indicate the purpose of the report and its intended audience. Clearly define the time period covered in the report. Then, explain what your project is, what its objectives are, and what the status of the project was at the time of the last reporting.

2. Project Description

In very short reports, the introduction might contain this section, but if it is under its own heading, readers who are familiar with the project can skip it. Someone unfamiliar with the project, however, needs summarized details such as purpose and scope of the project, start and completion dates, and names of parties involved.

3. Work Completed

This section follows the progress of the tasks presented in the proposal's schedule. This is the substance of the report. This section would be a project-tasks approach, a time-periods approach or a combined approach.

- Project-tasks approach: Focus on the tasks. Defined milestones can logically organize your discussion into this kind of structure. Also if you are working on a number of semi-independent tasks at the same time, this approach will work well.
- Time-periods approach: Focus on time. If a timeline (or deadline) is more important than milestones, then use this approach. Also, use it for projects with a simple linear structure.
- Combined approach: The two above approaches could be combined if, for example, under previous work, you break down what you have done by individual tasks. Or, under the tasks, you focus on what part is complete, what part is in progress, and what part is yet to come.

Your project will determine which of these three you use. If the problems encountered or

changes required are time-related, then use the time-periods approach to your advantage; likewise, if the problems or changes relate to specific tasks then use the project-tasks approach. Another item that may be included here is a summary of financial data. This last item could be contained in a table or appendix, or an independent section.

4. Problems Arisen

Progress reports are not necessarily for the benefit of only the client. As noted in the opening, snags are expected. Don't hide from them; explain what they are and how they might affect key areas of the job (such as timing, price or quality). Often, you the engineer or scientists benefit from the reporting because you can share or warn your client about problems that have arisen. If the problem occurred in the past, you can explain how you overcame it. This is least serious; in fact, you look good. If the problem is in front of you (now or in the future), explain how you hope to overcome it, if you can.

5. Changes in Requirements

Here, you record the changes to the project: milestones added, new requirements, or schedule changes (good or bad). Even if these changes have not affected the ultimate goal of the project, you need to tell the sponsor how problems have been accommodated.

Note: If changes are a direct result of problems encountered, sections 4 and 5 may be combined. This would lead to a modified organization: first problem and the change it required, then the next problem and change, and so on.

6. Work Scheduled

In this section, you discuss your plan for meeting the objectives of the project. In many ways, this section of a progress report is written in the same manner as the "Plan of Action" section of the proposal, except that now you have a better perspective for the schedule and cost than you did earlier. You can use one of those three approaches mentioned above (a project-tasks approach, a time-periods approach or a combined approach) according to your project in this section like in the section "Work Completed".

7. Overall Assessment of the Project

Since a progress report is not about a finished work, the conclusion needs only to give your professional opinion of how the project is going. Being unrealistically optimistic is as inappropriate as being unduly negative. Beware of promising early completion: a single setback can gobble up much time. Likewise, don't overreact if you are behind schedule. You may also gain time along the way. Far more significant for the engineer is to explain anything that may change the expected quality of the final product. Keeping in mind your purpose can help you focus here: Your goal is to enable the manager or sponsor to make informed decisions.

A Sample Progress Report

TO: Steven Zhang

Department of Computer Science, University of South California

FROM: Jane Li

SUBJECT: Progress on Scheme Debugging Report

DATE: July 20, 2009

This memo describes the progress I have made to date on my independent-study project to write a report on debugging in Scheme. In this memo, I review the nature of the project and describe work I have completed, work I am currently engaged in, and work I plan to complete by the end of the project. As I described in my memo of June 21, this project will result in a technical report whose purpose is to provide readers with practical information on developing and debugging programs in Scheme, supplementary to the material in your textbook, *An Introduction to Scheme and its Implementation*.

Project Description

The report is aimed at students in computer science (undergraduate and graduate) who have previous programming experience, but are new to Scheme. The information in this report is needed because readers who have developed programs using compilers for other languages may be unfamiliar with the approaches available with an interactive interpreter and debugger.

Project Scope

In my earlier memo, I proposed to cover the following high-level topics:

- Loading the debugging module into the interpreter
- Establishing break levels
- Applying back-trace
- Managing dependencies
- Saving and loading a customized heap image of the Scheme system
- Debugging local definitions
- Debugging native-code procedure calls
- Debugging when using functional programming style
- Program design and implementation strategies
- Using stubbed procedures
- Differences between RScheme and other Scheme systems

In my current outline, these are divided into three major parts, with an addendum for topic 11. The three parts are: (A) basic debugging procedures—topics 1-3, (B) advanced debugging procedures—topics 4-7, and (C) general program development strategies—topics 8-10.

Work Completed

I have completed first drafts of the sections in part A on loading the debugging module, break

levels, and apply-back-trace. I intend to make note of additional material for these sections while working on the later sections, if further background information is needed.

Present Work

I am currently working on the sections in part B. Since these sections are highly interrelated, I am working on them roughly in parallel. I am also currently researching information on other Scheme systems for section 11; I have located information on Gambit and DrScheme. I expect the current work to be completed by the end of this week, July 25th.

Future Work

Next, I will draft the sections in part C and the addendum on other Scheme systems. Finally, I will fully revise the entire draft, integrating further material where deficiencies have become evident during work on other sections. The final report will be ready for your review on August 6th.

Conclusion

Thus far, the project is proceeding well. I have not run into any major problems, nor do I anticipate any in the remaining work.

Exercises

1. What is this progress report mainly talking about?
2. Based on the project you are doing, write a similar progress report.

Testing the System

系统测试

Part 1 Listening & Speaking

► Dialogue: Software Testing

Kevin: It seems that we haven't spent enough time on talking about testing.

Sharon: True, but we have been thinking about it. In fact, more than thinking. Our testing strategy planned in the requirements phase, **takes an incremental view of** testing, beginning with the testing of individual program units, moving to testing designed to facilitate the integration of the units, and **culminating** with testing that **exercises** the constructed system.

Kevin: Yes. We have completed the design of the unit tests for each component of our system before coding begins, which is a preferred approach by agile programming. Even though we are not using Extreme Programming, we decided that it would be a good idea—a review of design information provides guidance for establishing **test cases** that are likely to **uncover** errors in each of the categories discussed earlier. Now, we can conduct our unit testing using these test cases.

Jason: During the integration testing phase, we can use some classical integration test strategies, such as top-down or bottom-up. Besides those used for traditional, hierarchically structured software, we can also use architecture-driven, which implies integrating the software components or subsystems based on identified functional **threads** according to the requirements specification.^[1]

Kevin: Well done. During the integration testing, we can develop a validation strategy at the same time. In this step, we should define testing required for validation and its criteria according to the software

[1] Replace with:

1. We can also use **black-box testing** when testing the input/output behavior of system.
2. We can also use **regression testing**

requirements specification, and information contained in its Validation Criteria section forms the basis for a validation testing approach.

Sharon: Maybe we have to perform some high-level testing such as recovery testing, security testing, **stress testing** and performance testing. Finally, we will coordinate **acceptance testing** with the customer.

Words

culminate

['kʌlmineɪt] v. 告终, 达到顶点

exercise

['eksəsaɪz] v. 执行

uncover

[ʌn'kʌvə] v. 揭开, 揭露

threads

[θredz] n. 各组成部分

Phrases

takes a view of	观察, 检查
test case	测试用例
stress testing	压力测试
acceptance testing	验收测试
black-box testing	黑盒测试
regression testing	回归测试

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

► Listening Comprehension: Software Testing

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- Which kind of approaches is also called “functional testing”?
 (A) White-box testing
 (B) Black-box testing
 (C) Grey-box testing
 (D) State-based testing
- Which kind of approaches does the mutation testing belong to?
 (A) White-box testing

- (B) Black-box testing
(C) Grey-box testing
(D) State-based testing
3. Which testing strategy is generally wrong according to this article?
(A) Using white-box testing for unit testing
(B) Using black-box testing for integration testing
(C) Using white-box testing for system testing
(D) Using black-box testing for acceptance testing

Words

verification

[ˌverɪfɪˈkeɪʃən] n. 确认, 核查

introduce

[ˌɪntrəˈdjuːs] v. 引入

deduce

[dɪˈdjuːs] v. 推论, 演绎出

mutation

[mjuːˈteɪʃən] n. 突变, 改变

mutant

[ˈmjuːtənt] n. 变体

Phrases

due to

由于

cause-effect

因果

state-based

基于状态的

grey-box testing

灰盒测试

in that

因为

white-box testing

白盒测试

➤ Dictation: Smoke Test

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

In electronics, the first time a circuit is attached to power, which will sometimes produce actual smoke if a design or wiring mistake has been made, is called “1 testing”. It refers to the first test made after repairs or first 2 to provide some assurance that the system under test will not catastrophically fail.

In software, the term smoke testing is metaphorical which describes the process of 3 code changes before the changes are 4 into the product's source tree. After code reviews, smoke testing is the most cost effective method for identifying and fixing 5

in software. It is designed to ____ 6 ____ that changes in the code function as ____ 7 ____ and do not ____ 8 ____ an entire build.

McConnell, the author of Code Complete, says that, “Whatever ____ 9 ____ strategy you select, a good approach to integrating the software is the ‘daily ____ 10 ____ and smoke test’.” Every file is compiled, linked, and combined into an executable program as a “build” every day, and the program is then **put through** a “smoke test,” a relatively simple check to exercise the entire system **from end to end**. It does not have to be ____ 11 ____, but it should be capable of ____ 12 ____ major problems that will keep the build from properly performing its function.

This simple process provides a number of significant ____ 13 ____ when it is applied on complex, time-____ 14 ____ software engineering projects. Firstly, it minimizes the integration ____ 15 ____ by ____ 16 ____ on all the code daily and ____ 17 ____ quality problems early. It also simplifies defect diagnosis and ____ 18 ____ because the software that has just been added to the build(s) is a probable cause of a newly discovered error. Besides, it supports easier progress ____ 19 ____ and improves **morale** when developers seeing ____ 20 ____ more of the product works every day with daily builds.

Words

catastrophically

[ˌkætəˈstrɒfɪkəli] adv. 灾难性地

metaphorical

[ˌmetəˈfɒrɪkəl] adj. 隐喻性的, 比喻性的

build

[bɪld] n. 编好的程序

morale

[məˈrɑ:l] n. 士气, 民心

Phrases

smoke test

冒烟测试

put through

完成

from end to end

从头到尾地

Part 2 Reading & Translating

➤ Section A: Software Testing

Testing is an activity performed for evaluating product quality, and for improving it, by identifying defects and problems. The view of software testing has evolved towards a more constructive one. Testing is **no longer** seen as an activity which starts only after the coding

phase is complete, with the limited purpose of detecting failures. Software testing is now seen as an activity which should encompass the whole development and maintenance process and is itself an important part of the actual product construction. Indeed, planning for testing should start with the early stages of the requirement process, and test plans and procedures must be systematically and continuously developed, and possibly refined, as development proceeds. These test planning and designing activities themselves constitute useful input for designers in **highlighting** potential weaknesses (like design **oversights** or contradictions, and **omissions** or ambiguities in the documentation).

Software testing is usually performed at different levels along the development and maintenance processes. **That is to say**, the target of the test can vary: a single module, a group of such modules (related by purpose, use, behavior, or structure), or a whole system. Three big test stages can be conceptually distinguished, namely Unit, Integration, and System. No process model is implied, nor are any of those three stages assumed to have greater importance than the other two.



Unit testing

Unit testing **verifies** the functioning in **isolation** of software pieces which are separately testable. Depending on the context, these could be the individual subprograms or a larger component **made of** tightly related units. Typically, unit testing occurs with access to the code being tested and with the support of debugging tools, and might involve the programmers who wrote the code.

Integration testing

Integration testing is the process of verifying the interaction between software components. Classical integration testing strategies, such as top-down or bottom-up, are used with traditional, hierarchically structured software. Modern systematic integration strategies are rather

architecture-driven, which implies integrating the software components or subsystems based on identified functional threads.

System testing

System testing is **concerned with** the behavior of a whole system and is usually considered appropriate for **comparing** the system **to** the non-functional system requirements, such as security, speed, accuracy, and reliability. External interfaces to other applications, utilities, hardware devices, or the operating environment are also evaluated at this level.

Other important objectives for testing include (but are not limited to) usability evaluation, and acceptance, for which different approaches would be taken. Note that the test objective varies with the test target; in general, different purposes being addressed at a different level of testing.

Acceptance/qualification testing

Acceptance testing checks the system behavior against the customer's requirements, however these may have been expressed; the customers **undertake**, or specify, typical tasks to check that their requirements have been met or that the organization has identified these for the target market for the software. This testing activity may or may not involve the developers of the system.

Alpha and beta testing

Before the software is **released**, it is sometimes given to a small, representative set of potential users for trial use, either **in-house** (alpha testing) or external (beta testing). These users report problems with the product. Alpha and beta use is often uncontrolled, and is not always referred to in a test plan.

Many techniques have been developed to perform testing, which attempt to “break” the program, by running one or more tests drawn from identified classes of executions **deemed** equivalent. [1] The leading principle underlying such techniques is to be systematic as possible in identifying a representative set of program behaviors; for instance, considering subclasses of the input domain, scenarios, states, and dataflow. It is difficult to find a **homogeneous** basis for classifying all techniques, and the one used here must be seen as a **compromise**. The classification is based on how tests are generated from the software engineer's **institution** and experience, the specification, the code structure, the (real or artificial) faults to be discovered, the field usage, or finally, the nature of the application. Sometimes these techniques are **classified as** white-box, also called glass-box, if the tests rely on information about how the software has been designed or coded, or as black-box if the test cases rely on the input/output behavior. One last category deals with combined use of two or more techniques. Obviously, these techniques are not used equally often by all **practitioners**.

Words**highlight**

['haɪlaɪt] v. 强调, 使突出

oversight

['əʊvəsait] n. 疏忽, 漏失

omission

[əʊ'mɪʃən] n. 遗漏

verify

['verɪfaɪ] v. 检验, 核实

isolation

[aɪsəʊ'leɪʃən] n. 隔离, 孤立

undertake

[ʌndə'teɪk] v. 着手做, 开始进行

release

[rɪ'li:s] v. 发布, 公开发行

in-house

[ɪn'hauz] adj. 内部的

deem

[di:m] v. 认为

homogeneous

[həməu'dʒi:njəs] adj. 相同特征的, 同性质的

compromise

['kɒmprəmaɪz] n. 妥协, 折中

institution

[ɪnstɪ'tju:ʃən] n. 习俗, 制度

practitioner

[præk'tɪʃənə] n. 从业者, 实践者

Phrases

no longer	不再
that is to say	即, 换句话说
be made of	用……制成
be concerned with	参与, 关于
compare to	把……比作
classify as	把……归类为

Complex Sentences

Original: Many techniques have been developed to perform testing, which attempt to “break” the program, by running one or more tests drawn from identified classes of executions deemed equivalent.

Translation: 很多技术已被开发用于执行测试, 通过运行一个或多个从已识别的、被认为是等价类的执行中获得的测试, 尝试“破坏”程序。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Test plan should initiate in the early stage of a software project.
- ___ 2. Developers of the system only need to participate in the first three testing activities.
- ___ 3. Top-down or bottom-up are two classical integration testing strategies.

- ___ 4. Test plan is helpful for preventing design defects.
- ___ 5. Glass-box techniques are used to test the inner structures of software.

II. Choose the best answer to each of the following questions.

1. Which statement is wrong about the software testing?
 - (A) Software testing should cover maintenance process besides development process.
 - (B) Test plan should be completed before coding.
 - (C) Test planning and designing activities can forecast potential weaknesses.
 - (D) The target of the testing can vary at different levels and phases.
2. Which of the following is right about the testing activities?
 - (A) Unit testing begins when all the units have been completed.
 - (B) System testing checks whether the system meets the customer's non-functional requirements.
 - (C) Acceptance testing checks whether the system meets the customer's functional requirements.
 - (D) Before the software is released, it is sometimes given to a small, random set of potential users for trial use.
3. Which of the following is right about the testing techniques?
 - (A) The testing techniques perform by running one or more identified classes of executions.
 - (B) The leading testing principle is to identify a representative set of program behaviors.
 - (C) All the testing techniques are classified based on a homogeneous standard.
 - (D) Black-box techniques do not care how the software has been designed or coded.

III. Translation

1. Testing is an activity performed for evaluating product quality, and for improving it, by identifying defects and problems.
2. Modern systematic integration strategies are rather architecture-driven, which implies integrating the software components or subsystems based on identified functional threads.

➤ Section B: GPGPU 通用计算图形处理器

GPGPU (general purpose computing on graphics processing units) is a methodology for high-performance computing that uses graphics processing units to **crunch** data. The characteristics of graphics algorithms that have enabled the development of extremely high-performance special purpose graphics processors **show up** in other HPC algorithms. This same special-purpose hardware can be put to use accelerating those algorithms as well.

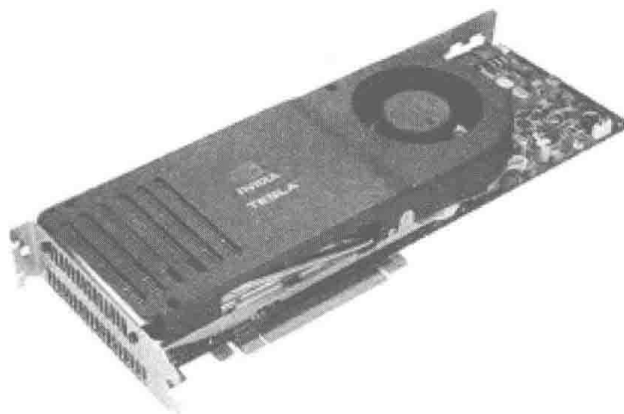
Algorithms well-suited to GPGPU implementation are those that exhibit two properties: they are data parallel and **throughput intensive**. Data parallel means that a processor can execute the operation on different data elements simultaneously. Throughput intensive means that the algorithm is going to

process lots of data elements, so there will be plenty to **operate on** in parallel. Taking advantage of these two properties, GPUs achieve extreme performance by incorporating lots (hundreds) of relatively simple processing units to operate on many data elements simultaneously.

Perhaps not surprisingly, **pixel**-based applications such as computer vision and video and image processing are very well suited to GPGPU technology, and for this reason, many of the commercial software packages in these areas now include GPGPU **acceleration**. Physical simulation applications, which often rely on numerical solutions to **PDEs** and on the heavy use of linear algebra operations, are also very well-suited to GPGPU technology. In the area of **rendering**, ray tracing (a computationally intensive algorithm for highly realistic image synthesis) uses GPGPU technology to compute ray-object intersections, making real-time ray tracing achievable.

GPU properties lead to very different processor architecture from traditional CPUs. CPUs devote a lot of resources (primarily chip area) to make single streams of instructions run fast, including **caching** to hide memory latency and complex instruction-stream processing (**pipelining**, out-of-order execution and speculative execution). GPUs, on the other hand, use the chip area for hundreds of individual processing elements that execute a single instruction stream on many data elements simultaneously. Memory latency is hidden by very fast context switching; when a memory fetch is issued while processing one subset of data elements, that subset is set aside in favor of another subset that is not waiting on a **memory reference**. [1]

GPUs can run certain algorithms anywhere from 10 to 100 or more times faster than CPUs — a huge advantage.



There are two disadvantages: 1) Gaining this **speedup** requires that algorithms are coded to reflect the GPU architecture, and programming for the GPU differs significantly from traditional CPUs. In particular, incorporating GPU acceleration into pre-existing codes is more difficult than just moving from one CPU family to another; a GPU-**savvy** programmer will need to **dive into** the code and make significant changes to critical components. 2) Incorporating GPU hardware into systems adds expense in terms of power consumption, heat production, and cost. Some job mixes may be served more economically by systems that maximize the number of

CPUs that can be **brought to bear**.

GPGPU computing is making a significant impact on high-performance computing in a wide range of application domains. Widely used HPC codes in areas including weather forecasting, molecular dynamics, and fluid-flow are being updated to incorporate GPGPU acceleration. Popular commercial scientific and engineering applications now provide GPGPU acceleration, **notably** MATLAB and ANSYS; open-source systems including AMBER, LAMMPS, NAMD; and Gromacs are now using GPGPU technology. Even more significant is the wide acceptance of GPGPU technology in new code development for **leading-edge** scientific and engineering research.

High-performance computing has long relied on parallel systems to achieve the performance required by leading-edge science and engineering research. **With the advent of** GPGPU technology, the capability of single nodes is greatly enhanced, but still does not provide sufficient performance for current HPC applications, which currently rely on systems of tens of thousands of individual CPUs working **in concert** to achieve the necessary performance. [2]

Words

crunch

[krʌntʃ] v. 处理, [俚]运行

accelerate

[ək'seləreɪt] v. (使)加快, (使)增进

well-suited

['wel'sutɪd] adj. 适当的, 便利的

throughput

['θruːput] n. 吞吐量, 生产量

intensive

[ɪn'tensɪv] adj. 密集的, 加强的, 集中的

pixel

['pɪksl] n. (显示器或电视机图像的)像素

acceleration

[ək'selə'reɪʃən] n. 加速, 加速升级

render

['rendə] v. 着色, 渲染

cache

[kæʃ] v. 把(数据)存入高速缓冲存储器, 高速缓存

pipeline

['paɪpleɪn] v. 传递

speedup

['spiːdʌp] n. 加速

savvy

['sævi] adj. 有经验的, 有见识的, 具有实际知识的

notably

['nəʊtəbli] adv. 特别, 尤其

Phrases

show up

显而易见, (使)显现出来

operate on

产生作用

memory reference

存储器参量, 存储器参考, 内存引用

dive into	投入, 钻研
bring to bear	应用, 使用
leading-edge	领先优势的
with the advent of	随着……到来
in concert	异口同声地, 齐声

Abbreviations

GPGPU	General Purpose computing on Graphics Processing Units	通用计算图形处理器
HPC	High Performance Computing	高性能计算
PDE	Partial Differential Equations	偏微分方程

Complex Sentences

[1] **Original:** Memory latency is hidden by very fast context switching; when a memory fetch is issued while processing one subset of data elements, that subset is set aside in favor of another subset that is not waiting on a memory reference.

Translation: 内存延迟会被非常快速的上下文转化所掩盖: 当数据元素的一个子集在处理过程中需要进行存储器读取时, 这个子集会被搁置并让另一个无须在存储器参量中等待的子集运行。

[2] **Original:** With the advent of GPGPU technology, the capability of single nodes is greatly enhanced, but still does not provide sufficient performance for current HPC applications, which currently rely on systems of tens of thousands of individual CPUs working in concert to achieve the necessary performance.

Translation: 随着 GPGPU 技术的实现, 单个节点的能力大大增强, 但仍然不能为当前的 HPC 应用提供足够的性能。这些应用目前仍依赖于数万个单独 CPU 同时工作的系统, 来获取所需要的性能。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. CPUs can run certain algorithms anywhere from 10 to 100 or more times faster than GPUs.
- ___ 2. GPGPU computing is not making a significant impact on high-performance computing in a wide range of application domains.
- ___ 3. Algorithms well-suited to GPGPU implementation are those with two properties, which

are data parallel and throughput intensive.

- ___ 4. One of the disadvantages of GPU is that incorporating GPU hardware into systems adds expense in terms of power consumption, heat production, and cost.
- ___ 5. GPU properties lead to the same processor architecture as traditional CPUs.

II. Choose the best answer to each of the following questions.

1. Which statement is wrong about the GPU?
 - (A) GPU properties lead to very different processor architecture from traditional CPUs.
 - (B) Incorporating GPU acceleration into pre-existing codes is more difficult than just moving from one CPU family to another.
 - (C) Data parallel means that a processor can execute the operation on different data elements sequentially.
 - (D) Throughput intensive means that the algorithm is going to process lots of data elements, so there will be plenty to operate on in parallel.
2. How many disadvantages of GPU have been mentioned in this article?
 - (A) One
 - (B) Two
 - (C) Three
 - (D) Four
3. Which application now provides GPGPU acceleration?
 - (A) Microsoft Word
 - (B) Adobe Photoshop
 - (C) Google Docs
 - (D) MATLAB

III. Translation

1. Perhaps not surprisingly, pixel-based applications such as computer vision and video and image processing are very well suited to GPGPU technology, and for this reason, many of the commercial software packages in these areas now include GPGPU acceleration.
2. Gaining this speedup requires that algorithms are coded to reflect the GPU architecture, and programming for the GPU differs significantly from traditional CPUs.

Part 3

Simulated Writing: Software Test Specification

The test plan describes an overall breakdown of testing into individual tests that address specific items. For each such individual test, we write a test specification and evaluation. The specification begins by listing the requirements whose satisfaction will be demonstrated by the test. Referring to the requirements documents, this section explains the test's purpose.

One way to view the correspondence between requirements and tests is to use a table or chart. The system functions involved in the test are enumerated in the table. The performance tests can be described in a similar way. Instead of listing functional requirements, the chart lists requirements related to speed of access, database security, and so on.

Often, an individual test is really a collection of smaller tests, the sum of which illustrates requirements satisfaction. In this case, the test specification shows the relationship between the smaller tests and the requirements.

Each test is guided by a test philosophy and adopts a set of methods. However, the philosophy and methods may be constrained by other requirements and by the realities of the test situation. The specification makes these test conditions clear. Among the conditions may be some of the following:

- Is the system using actual input from users or devices, or are special cases generated by a program or surrogate device?
- What are the test coverage criteria?
- How will data be recorded?
- Are there any timing, interface, equipment, personnel, database, or other limitations on testing?
- If the test is a series of smaller tests, in what order are the tests to be performed?

If test data are to be processed before being evaluated, the test discusses the processing. For instance, when a system produces large amounts of data, data reduction techniques are sometimes used on the output so that the result is more suitable for evaluation.

Accompanying each test is a way to tell when the test is complete. Thus, the specification is followed by a discussion of how we know when the test is over and the relevant requirements have been satisfied. For example, the plan explains what range of output results will meet the requirement.

The evaluation method follows the completion criteria. For example, data produced during testing may be collected and collated manually and then inspected by the test team. Alternatively, the team could use an automated tool to evaluate some of the data and then inspect summary reports or do an item-by-item comparison with expected output.

Software Test Specification Template

1. Introduction

This section provides an overview of the entire test document. This document describes both the test plan and the test procedure.

1.1 Goals and objectives

Overall goals and objectives of the test process are described.

1.2 Statement of scope

A description of the scope of software testing is developed. Functionality/features/behavior to be tested is noted. In addition any functionality/features/behavior that is not to be tested is also noted.

1.3 Major constraints

Any business, product line or technical constraints that will impact the manner in which the software is to be tested are noted here.

2. Test plan

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

2.1 Software to be tested

The software to be tested is identified by name. Exclusions are noted explicitly.

2.2 Testing strategy

The overall strategy for software testing is described.

2.2.1 Unit testing

The strategy for unit tested is described. This includes an indication of the components that will undergo unit tests or the criteria to be used to select components for unit test. Test cases are not included here.

2.2.2 Integration testing

The integration testing strategy is specified. This section includes a discussion of the order of integration by software function. Test cases are not included here.

2.2.3 Validation testing

The validation testing strategy is specified. This section includes a discussion of the order of validation by software function. Test cases are not included here.

2.2.4 High-order testing

The high-order testing strategy is specified. This section includes a discussion of the types of high order tests to be conducted, the responsibility for those tests. Test cases are not included here.

2.3 Testing resources and staffing

Specialized testing resources are described and staffing is defined. The role of any ITG is also defined.

2.4 Test work products

The work products produced as a consequence of the testing strategy are identified.

2.5 Test record keeping

Mechanisms for storing and evaluating test results are specified.

2.6 Test metrics

A description of all test metrics to be used during the testing activity is noted here.

2.7 Testing tools and environment

A description of the test environment, including tools, simulators, specialized hardware, test files, and other resources is presented here.

2.8 Test schedule

A detailed schedule for unit, integration, and validation testing as well as high order tests is described.

3. Test Procedure

This section describes as detailed test procedure including test tactics and test cases for the software.

3.1 Software to be tested

The software to be tested is identified by name. Exclusions are noted explicitly.

3.2 Testing procedure

The overall procedure for software testing is described.

3.2.1 Unit test cases

The procedure for unit testing is described for each software component (that will be unit tested) is presented. This section is repeated for all components i.

3.2.1.1 Stubs and/or drivers for component i

3.2.1.2 Test cases component i

3.2.1.3 Purpose of tests for component i

3.2.1.4 Expected results for component i

3.2.2 Integration testing

The integration testing procedure is specified.

3.2.2.1 Testing procedure for integration

3.2.2.2 Stubs and drivers required

3.2.2.3 Test cases and their purpose

3.2.2.4 Expected results

3.2.3 Validation testing

The validation testing procedure is specified.

3.2.3.1 Testing procedure for validation

3.2.3.2 Expected results

3.2.3.3 Pass/fail criterion for all validation tests

3.2.4 High-order testing (also known as system testing)

The high-order testing procedure is specified. For each of the high order tests specified below, the test procedure, test cases, purpose, specialized requirements and pass/fail criteria are specified. It should be noted that not all high-order test methods noted in Sections 3.2.4.n will be conducted for every project.

3.2.4.1 Recovery testing

3.2.4.2 Security testing

3.2.4.3 Stress testing

3.2.4.4 Performance testing

3.2.4.5 Alpha/beta testing

3.2.4.6 Pass/fail criterion for all validation tests

3.3 Testing resources and staffing

Specialized testing resources are described and staffing is defined. The role of any ITG is also defined.

3.4 Test work products

The work products produced as a consequence of the testing procedure are identified.

3.5 Test record keeping and test log

Mechanisms for storing and evaluating test results are specified. The test log is used to maintain a chronological record of all tests and their results.

PS: This template can be tailored according to the different projects.

A Sample Software Test Specification

Software Test Specification For Web Publishing System Version 1.0

Victor Lu (Team Leader)
Jack Liang
Jason Wang
Remonna Zeng

May 31, 2013

★ *Table of Contents*

Table of Contents

1. Introduction
 - 1.1 Purpose
 - 1.2 Scope of Testing
 - 1.3 Glossary
 - 1.4 References
 - 1.5 Overview of Document
2. Verification Testing
 - 2.1 Unit Relationships
 - 2.2 Unit Testing
Test Unit: AuthorSearch
 - 2.3 Integrative Testing
Test Unit: On-Line Journal (JournalHomePage)
3. System (In-House Validation) Testing
 - 3.1 Use Case: Search Article
 - 3.2 Use Case: Updnte Arficle Status
4. Stakeholder Testing
 - 4.1 Acceptance Testing
 - 4.2 Beta Testing

1. Introduction

1.1 Purpose

This document is part of the Software Verification and Validation Plan.

Its primary audiences are the development team, the in-house quality assurance team, and the stakeholders. An important secondary audience is the maintenance team who will need to correct, modify, and enhance this product.

1.2 Scope of Testing

This document will be used by the development team to conduct verification and validation tests for the product. It also details the Acceptance Testing that will be done and the Beta Testing procedures. This document contains only two sample verification tests and two sample use case validation tests in accordance with the class assignment.

1.3 Glossary

Term	Definition
Acceptance Testing	Testing done in the stakeholder environment. It may be done by the stakeholder, the development team, or an outside team as agreed to by the stakeholder
Beta Testing	The preliminary usage of the system by the stakeholder
Validation Testing	Testing done to assure that the product satisfies all of the needs of the stakeholder and the requirements listed in the Software Requirements Specification
Verification testing	Testing done to assure that the modules of the project perform as required by the Software Design Specification

1.4 References

Software Requirements Specification for Web Publishing System,2013

Software Design Specification for Web Publishing System,2013

1.5 Overview of Document

In the next section, the verification tests for two of the major design entities are given.

In the following section, sample validation tests for some of the major use cases as required by a previous assignment are given.

The final chapter outlines the installation and acceptance testing procedure and gives the conditions for beta testing the software.

2. Verification Testing

2.1 Unit Relationships

Relationships other than on tables:

Unit Name	Depends on
User (interface)	On-Line Journal (Journal Home Page)
Journal Home Page	AuthorSearch
Journal Home Page	CategorySearch
Journal Home Page	KeywordSearch
Editor (interface)	Article Manager
Article Manager	PeopleManager
PeopleManager	HS Database (interface)
Article Manager	ArtMgr
Article Manager	Publishetr

Table dependencies are as follows:

Unit Name	Depends on Table
AuthorSearch	Author
AuthorSearch	Article
CategorySearch	Article
CategorySearch	Category
KeywordSearch	Article
PeopleManager	Reviewer
PeopleManager	Reviewer-Relationship
PeopleManager	Author
PeopleManager	Author-Relationship
ArtMgr	Reviewer
ArtMgr	Author
ArtMgr	Author-Relationship
ArtMgr	Reviewer-Relationship
ArtMgr	Category-Relationship
ArtMgr	Active Article
Publisher	Active Article
Publisher	Article (Journal Database)

2.2 Unit Testing

Test Unit: AuthorSearch

Cross-Reference: SDS 3.1.2

Driver: None needed

At least five articles must be in the database. The following will suffice.

Authors	Title	Categories	Abstract	Body
Dean, James Nash, Ogden Paddle, Wheel	Ways to Avoid Testing	Humor, Satire	This article involves ridiculous ideas about mud.	<insert block of random text here>
Dean, James	Testing is Fun	Satire	Ridiculous is a manner of speaking.	< insert different block of random text here>
Dean, James Double, Mint	Nonsense I Have Known	Romance	No keyword in common.	< insert different block of random text here>
Paul, Peter	Apostles of Testing	Software Engineering	Speaking of mud and mud.	< insert different block of random text here>
Double, Mint Nash, Ogden	Software is Fun to Create	Software Engineering Romance	Ridiculous mud ideas.	< insert different block of random text here>

Method: Use interactive input.

Each author must be tested for each article by that author. Continue each until the article is downloaded (There are nine tests to be performed).

The test is successful when the correct article is downloaded each time.

2.3 Integrative Testing

Test Unit: On-Line Journal (Journal Home Page)

Cross-Reference: SDS 3.1.1

Driver: None needed

Method: This unit is partly tested by testing the Use Case Search Article (see 3.1 below).

In addition, we must test the E-mail capability. Select to send E-mail then send a message to the editor of the journal attaching a text file.

The test is successful if the text file can be successfully downloaded by the editor.

3. System (In-House Validation) Testing

3.1 Use Case: Search Article

Reference: SRS, Section 3.2.1.

Environment: Access the database from a computer on the Internet other than the one on which the database resides.

Since this use case uses the Online Journal database exclusively, we will create a copy of that database by entering data directly into tables. The following is a minimum of the data that must be available for testing:

At least five articles must be in the database. The following provides a variety of options to be tested.

Authors	Title	Categories	Abstract	Body
Dean, James Nash, Ogden Paddle, Wheel	Ways to Avoid Testing	Humor, Satire	This article involves ridiculous ideas about mud.	<insert block of random text here>
Dean, James	Testing is Fun	Satire	Ridiculous is a manner of speaking.	< insert different block of random text here>
Dean, James Double, Mint	Nonsense I Have Known	Romance	No keyword in common.	< insert different block of random text here>
Paul, Peter	Apostles of Testing	Software Engineering	Speaking of mud and mud.	< insert different block of random text here>
Double, Mint Nash, Ogden	Software is Fun to Create	Software Engineering Romance	Ridiculous mud ideas.	< insert different block of random text here>

Method: Use interactive input.

There are three functions to be tested, lookup by author, category and keyword. These are to be tested in random order.

Each author must be tested. Continue each until the article is downloaded.

Each category must be tested. Continue each until the article is downloaded.

Each noun in the Abstract column is to be tested as a keyword. Continue each until the article is downloaded. Try words not contained in the abstract. Make sure that duplicate words do not produce duplicate responses. Make sure case is not significant.

Periodically, abandon a search in the middle and restart.

Results Expected: The test is successful if no problems are encountered.

3.2 Use Case: Update Article Status

Reference: SRS, Section 3.2.6.

Environment: This test will be performed on the Article Manager database. The use case *Enter Communication* must be implemented prior to validation testing of this use case.

At least five articles must be in the database at various status levels. The data listed in Section 2.1 can be used with the addition of status values.

Method: Use interactive input.

There are five functions to be tested, add (or remove) categories, correct typographical errors, remove a reviewer, enter an updated article, and enter a review. After each action is performed, inspect the database to see if the action has occurred.

Add a new category for an article.

Add two new categories for an article.

Delete a category for an article.

Attempt to delete the last category for an article. This should not be allowed.

Modify an author's name.

Modify a title.

Modify an abstract.

Remove a reviewer. If all the other reviews are entered, this should change the article's status accordingly.

Attempt to remove the last reviewer. This should not be allowed.

Enter an updated article.

Enter a review for an article. This should change the reviewer's status accordingly and if that is the last outstanding review, change the article's status accordingly.

Results Expected: The test is successful if no problems are encountered.

4. Stakeholder Testing

4.1 Acceptance Testing

The development team will preload the system with at least five articles in various stages of approval based on information provided by the stakeholder. The system will then be installed on the stakeholder's computer. The stakeholder will enter new information and update the preloaded information to test the system. This includes publishing accepted articles to the Online Journal and accessing these from a remote location by multiple simultaneous users.

Any discrepancies between the behavior described in the SRS and the actual system will be corrected and the system will be retested.

4.2 Beta Testing

Any problems detected by the stakeholder within six months of installation will be reported to the development team. After correction, a further six month period for beta testing will ensue under the same regulations. All costs for this work are covered by the original contract.

Exercises

1. What is the primary purpose of the tests mentioned in this software test specification?
2. How many functions are to be tested?
3. What dependencies will the test team need in order to fully conduct the tests?

Delivering the System

系统交付

Part 1 Listening & Speaking

► Dialogue Software Deployment

Mr. White: Welcome to our hotel! I really appreciate ^[1] your efforts for our Four Seasons Hotel Management Information System.

Kevin: It's our pleasure. Today, we'll deliver the software to you, including all tested program files, data files and additional documentation for user, such as a user guide and an operator manual. We have also added hypertext "help" files and a troubleshooting guide in our software.

Mr. White: It sounds wonderful! So can we run the system after deployment right away?

Kevin: No problem. But I suggest we test the software ^[2] in a small group of representative users first. After ensuring it runs normally, we can deploy the system in the whole hotel. In this way, those users can first run the software and find problems in practice and the impact will be controlled to a smaller scope and the problems can be solved in time. This is also a common way in software deployment.

Mr. White: Really? Well, I think it is a good idea. But how much of the scope does it fit? We have 100 staffs and 20 computers in total.

Sharon: Maybe one third of the information desk in your hotel is ok. In my opinion, information desks deal with most of hotel daily business, and can expose potential problems effectively and efficiently.

Mr. White: Ok. I completely **agree to** this suggestion, and will arrange the equipment and personnel as soon as possible. In addition, I am afraid we would need some timely support from you if some problems are discovered.

[1] Replace with:

1. I am very **grateful for**
2. I am very thankful for

[2] Replace with:

1. I suggest we conduct a beta testing
2. I suggest we conduct a test run.

Kevin: Don't worry about it. We will provide installation and **start-up** assistance, and as a support group, we will ensure troubleshooting assistance for you until the system runs normally for half a year.

Jason: Here is our contact list with the phone numbers and E-mail addresses. And we need some contact information of your personnel who are in charge of the system in your hotel as well, so as to communicate in a timely manner.

Mr. White: No problem. I will send you an E-mail listing our contact information within 2 days.

Kevin: On this aspect, furthermore, I think we had better formally establish problem-logging and feedback mechanism, which includes what the feedback process is, how you log problems and report them to us, and which form to use, paper and/or electronic, and so on. In this way, we can collect and log your feedback, communicate and assess it with you to determine the modification plan together.

Mr. White: That's right. Let's go on please.

Words

troubleshoot

['trʌblʃu:t] v. 故障检修, 问题解答

log

[lɒg] v. 记录

start-up

['stɑ:tʌp] n. 开始, 启动

Phrases

agree to

同意

be grateful for

为……而感谢

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ Listening Comprehension: Software Delivery

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

1. Which is the most appropriate order of steps in the software delivery stage?

(A) Fixing, delivery, feedback, support, and modification

(B) Delivery, fixing, support, feedback, and modification

- (C) Fixing, support, delivery, feedback, and modification
(D) Delivery, fixing, feedback, support, and modification
2. Which point is not correct on support before the software delivery according to this article?
(A) Support should be considered after the software delivery because the possible troubles are unpredictable during the practical running.
(B) The software team can conduct a categorical assessment of the kinds of support requested by an appropriate record keeping mechanisms.
(C) Support documents should be provided on the software delivery.
(D) A support regime must be established before the software is delivered.
3. Which point is correct on feedback after the software delivery according to this article?
(A) Feedback should be responded and resolved in the present delivered increment immediately and entirely.
(B) Feedback should not be answered by the software team anymore because the project has finished since the software delivery.
(C) Feedback allows the customer to suggest changes that have business value and provide the input for the next iterative software engineering cycle for the software team.
(D) Feedback is undesirable because of its resulting in some uncontrolled requirements and unexpected changes.

Words

cheer

[tʃiə] n. 欢呼, 喝彩

regime

[reiʒi:m] n. 体制, 体系

categorical

[ˌkætiˈɡɒrikəl] adj. 分类的, 按类别的

Phrases

put into use

应用, 使用

➤ Dictation: Bug & Debugging

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

The word “bug” has been used to describe a “bogyman” ever since the fourteenth century. In computers, it was popularly ____1____ to Admiral Dr. Grace Hopper, the inventor of COBOL, who ____2____ the first computer bugs in the 1940s. **Albeit** not the **flying** kind, software ____3____ manifest themselves in a variety of ways, from misunderstood requirements to coding ____4____.

Having been used as a term in **aeronautics** before 5 the world of computers, the word “debugging” is first used in three papers from 1952 ACM National Meetings. A software engineer, evaluating the results of a test, is often **confronted** with a “**symptomatic**” 6 of a software problem. That is, the 7 manifestation of the error and the 8 cause of the error may have no obvious 9 to one another. The poorly understood **mental** process that 10 a symptom to a cause is debugging. In detail, in computers, debugging is the process of 11 and fixing bugs in computer program code or the engineering of a hardware 12. It is a necessary process in almost any new software or hardware development process, whether a 13 product or an enterprise or personal 14 program.

No one writes perfect software, so it's a **given** that debugging will **take up** a major 15 of your days. In his book Debugging, David, J Agans provides the 9 Indispensable Rules for Finding Even the Most **Elusive** Software and Hardware Problems as follow: Understand the system, make it 16, 17 thinking and look, 18 and conquer, change one thing at a time, keep an **audit** 19, check the **plug**, get a 20 view, and if you didn't fix it, it **ain't** fixed.

Words

bogeyman

['bəʊɡimæn] n. 可怕（或可憎）的人或物，妖怪，精灵

albeit

[ɔ:l'bi:t] conj. 虽然

flying

['flaɪɪŋ] adj. 会飞的，飞行的

manifest

['mænɪfɛst] v. 显示，表明

aeronautics

[.ɛərə'no:tiks] n. 航空学

confront

[kən'frʌnt] v. 面对，碰到

symptomatic

[.sɪmptə'mætɪk] adj. 有症状的

mental

['mentl] adj. 智力的，脑力的

given

['ɡɪvn] n. 假设

elusive

[i'lju:sɪv] adj. 难于被发现的

audit

['ɔ:dit] n. 审核，查证

plug

[plʌɡ] n. 插头，插销

ain't

[eɪnt] prep. (= are not, am not, is not) 不是

Phrases

take up

占（空间、时间）

Part 2 Reading & Translating

➤ Section A: Software Maintenance

System development is complete when the system is operational, that is, when the system is being used by users in an actual production environment. Any work done to change the system after it is operational is considered to be maintenance. Many people think of software system maintenance as they do hardware maintenance: repair or prevention of broken or improperly working parts. However, software maintenance cannot be viewed in the same way. Let us see why.

One goal of software engineering is developing techniques that define a problem exactly, design a system as a solution, implement a correct and efficient set of programs, and test the system for faults. This goal is similar for hardware developers: producing a reliable, fault-free product that works according to specification. Hardware maintenance in such a system concentrates on replacing parts that **wear out** or using techniques that **prolong** the system's life. However, while do **constructs** do not wear out after 10,000 loops, and **semicolons** do not **fall off** the ends of statements. Unlike hardware, software does not **degrade** or require **periodic** maintenance. Thus, software systems are different from hardware, and we cannot use hardware **analogies** successfully the way we can for other aspects of software engineering.

The biggest difference between hardware and software systems is that software systems are built to **incorporate** change. **Except for** the simplest cases, the systems we develop are evolutionary. That is, one or more of the system's defining characteristics usually change during the life of the system.

Software systems may change not just because a customer makes a decision to do something in a different way, but also because the nature of the system itself changes. For example, consider a system that computes payroll deductions and issues paychecks for a company. The system is dependent on the tax laws and regulations of the city, state or province, and country in which the company is located. If the tax laws change, or if the company moves to another location, the system may require modification. Thus, system changes may be required even if the system has been working acceptably in the past.

When we develop systems, our main focus is on producing code that implements the requirements and works correctly. At each stage of development, our team continually refers to work produced at earlier stages. The design components are **tied to** the requirements specification, the code components are **cross-referenced** and reviewed for compliance with the design, and the tests are based on finding out whether functions and constraints are working according to the requirements and design. Thus, development involves looking back in a careful, controlled way.

Maintenance is different. As **maintainers**, we look back at development products, but also at the

[illegible]

Maintenance focuses on four major aspects of system evolution simultaneously:

- Often, a separate group of analysts, programmers, and designers (sometimes including one or two members of the development team) is designated as the maintenance team. A **fresh**, new team may be more objective than the original developers.

Maintaining a system involves all team members. Typically, users, operators, or customer representatives **approach** the maintenance team with a comment or problem. The analysts or programmers determine which parts of the code are affected, the impact on the design, and the likely resources (including time and effort) to make the necessary changes. The team is **involved in** many activities:

- (1) understanding the system
- (2) locating information in system documentation
- (3) keeping system documentation up to date
- (4) extending existing functions to accommodate new or changing requirements
- (5) adding new functions to the system
- (6) finding the source of system failures or problems
- (7) locating and correcting faults
- (8) answering questions about the way the system works
- (9) restructuring design and code components
- (10) rewriting design and code components
- (11) deleting design and code components that are no longer useful
- (12) managing changes to the system as they are made

In addition, maintenance team members work with users, operators, and customers. First, they try to understand the problem as expressed in the user's language. Then, the problem is transformed into a request for modification. The change request includes a description of how the system works now, how the user wants the system to work, and what modifications are needed to produce the changes. Once design or code is modified and tested, the maintenance team retrains the user, if necessary. Thus, maintenance involves interaction with people as well as with the software and hardware.

Words

prolong

[prə'lɒŋ] v. 延长

construct

[kən'strʌkt] n. 结构体

loop

[lu:p] n. 循环

semicolon

['semi'kəʊlən] n. 分号

degrade

[di'greɪd] v. 降级, 退化

periodic

[piəri'ɒdɪk] adj. 周期的, 定期的

analogy

[ə'nælədʒi] n. 类比, 类似

incorporate

[ɪn'kɔ:pəreɪt] v. 加入, 包含

cross-reference

['krɒsrefərəns] v. 前后对照

maintainer

[men'teɪnə] n. 维护人员

anticipate

[æn'tɪsɪpeɪt] v. 预期

smoothly

['smu:ðli] adv. 平稳地

intimate

['ɪntɪmɪt] adj. 精通的, 基本的

perfect

['pə:fɪkt] v. 使完善, 改进

fresh

[freʃ] adj. 新到的, 无经验的

approach

[ə'prəʊtʃ] v. 提议, 同……接洽

Phrases

wear out	用坏
fall off	脱落, 掉队
except for	除……之外
be tied to	被束缚
be involved in	参与

Complex Sentences

Original: we look back at development products, but also at the present by establishing a working relationship with users and operators to find out how satisfied they are with the way the systems works.

Translation: 我们不但要回头看开发的产品, 而且要看眼前的工作, 通过与用户和操作人员建立一种工作关系以发现他们是否满足于系统工作的方式。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Maintenance begins with any work done to change the system after it is operational.
- ___ 2. Software maintenance is similar to hardware maintenance.
- ___ 3. Maintenance activities are similar to those of development.
- ___ 4. Often, a maintenance team involves analysts, programmers, and designers.
- ___ 5. Maintenance requires the cooperation with users, operators, and customers.

II. Choose the best answer to each of the following questions.

1. Which of the following does not pertain to the characteristics of software maintenance?
 - (A) It does not require to be performed periodically.
 - (B) It has a broader scope, with more to track and control than development.
 - (C) It concentrates on replacing parts that wear out or using techniques that prolong the system's life.
 - (D) It involves interaction with people as well as with the software and hardware.
2. Which statement is wrong about the change of software?
 - (A) It derives from customers' new decisions to do something in different ways.
 - (B) A changing business need may bring some functional changes of software.
 - (C) A changing hardware, software, or interfaces need may bring some system changes of software.
 - (D) It can be resulted from the change of the natural of the system itself.

3. Which statement is wrong about the people who perform maintenance?
- (A) Users, operators, or customer representatives are also involved although they do not belong to the maintenance team.
 - (B) Programmers play a much larger role in maintenance than they did in development.
 - (C) The original group of programmers is more appropriate for maintenance than a fresh new team.
 - (D) Maintainers must look back, at the present, but also forward.

III. Translation

1. The design components are tied to the requirements specification, the code components are cross-referenced and reviewed for compliance with the design, and the tests are based on finding out whether functions and constraints are working according to the requirements and design.
2. The analysts or programmers determine which parts of the code are affected, the impact on the design, and the likely resources (including time and effort) to make the necessary changes.

➤ Section B: Bitcoin 比特币

Bitcoin is an Internet-based digital currency that uses strong **cryptography** to make sure nobody is duplicating money. This allows it to work without relying on any central government or management, which tends to **appeal to** libertarians, **anarcho-capitalists**, and technophiles.

The currency needs lots of computers to process and record transactions. The solution to this is “mining,” in which Bitcoin users run software on their computers to do all the necessary work. Every time someone successfully “finishes” a work package (which becomes more difficult every time anyone finishes one), they receive Bitcoins in return. This provides an **incentive** to keep the currency running, but also attracts a lot of **prospectors** and **speculators** looking for easy money, and **scammers** who consider them **suitably exploitable suckers**.

The interesting bit about Bitcoin is that it is entirely decentralized; there is no central bank backing. Previous virtual currencies, such as E-Gold, Flooz, Beenz, Lindens or WoW gold, have always had an organization behind them. This lack of central bank backing means that were governments to try to do something about it, they would not have a central point of attack.

Bitcoins are entirely imaginary currency (*i.e.*, they have no use-value), but not particularly more so than US dollars, and could be a general currency if 300 million people similarly behaved as though it was one, *i.e.*, would do work in exchange for it. [1] Its biggest problem as an exchange medium is that it is not widely accepted, and that trading is thus very thin indeed.



There is also the matter of built-in **deflation**. As more and more Bitcoins get **mined**, it requires more and more processing power to mine new ones. Also, if your wallet file is deleted, your Bitcoins are gone **for good**.

The processing of Bitcoin transactions is secured by servers called Bitcoin miners. These servers communicate over an Internet-based network and confirm transactions by adding them to a **ledger** which is updated and **archived** periodically using **peer-to-peer** file sharing technology. In addition to archiving transactions, each new ledger update creates some newly **minted** Bitcoins. The number of new Bitcoins created in each update is **halved** every 4 years until the year 2140 when this number will **round down** to zero. At that time no more Bitcoins will be added into circulation and the total number of Bitcoins will have reached a maximum of 21 million Bitcoins. To **accommodate** this limit, each Bitcoin is subdivided down to eight decimal places; forming 100 million smaller units called satoshis.

“Babbage” at The Economist takes it seriously (or at least **refrains from** just **poking fun at** it) and finds it quite interesting. Other economists have criticized the idea (to the point of calling it a **scam**), citing inherent design problems. Paul Krugman refrains from making fun of it, but considers it a reimplementaion of the gold standard, with the economic problems that implies. About 25% of the European Central Bank’s report on “Virtual Currency Schemes” is about Bitcoin.

You can buy actual stuff with Bitcoins! Mostly Internet services, **geek** toys, illegal drugs, and, of course, Bitcoin mining hardware. And actually useful things like beer. To allow payment with a high volatility currency like Bitcoin, it is common for merchants to price their goods in **fit currency**, but receive payment via Bitcoin converted at current market rates.

Words

cryptography

[krip'tɒgrəfi] n. 密码使用法, 密码系统, 密码术

libertarian

[libə'teəriən] n. (思想或行动等的)自由论者

technophile

['teknəu,faɪl] n. 技术爱好者

incentive

[in'sentiv] n. 刺激, 诱因, 鼓励, 动机

prospector

['prɒspektə] n. 勘探者, 探矿者

speculator

['spekjuleɪtə] n. 投机商人

scammer

['skæmə] n. [美俚] 骗子, 诈骗犯

suitably

['sju:təbli] adv. 相称地, 适当地, 适宜地

exploitable

[ɪks'plɔɪjəbl] adj. 有利可图的, 可开发的, 可利用的

sucker

['sʌkə] n. 对……入迷的人, 难以抗拒……的人

thin

[θɪn] adj. 稀少的

deflation

[di'fleɪʃən] n. 通货紧缩, 降阶

mine

[maɪn] v. 开采

ledger

['ledʒə] n. 总账, 账簿

archive

['ɑ:kəɪv] v. 存档

peer-to-peer

[pi: tu: pi:] adj. 对等

mint

[mɪnt] adj. 崭新的, 完美的, 未损坏的

halve

[hɑ:v] v. 把……分成两半, 把……减半

accommodate

[ə'kɒmədeɪt] v. 容纳

scam

[skæm] n. 骗局, 诡计

geek

[gi:k] n. 对计算机痴迷的人

Phrases

Bitcoin	点对点 (peer-to-peer) 基于网络的匿名数字货币
appeal to	对……有吸引力
anarcho-capitalist	无政府资本主义者
for good	永远
round down	四舍五入
refrains from	制止, 控制, 克制不要
poke fun at	取笑, 调笑, 作弄, 戏弄
fit currency	可使用的通货

Complex Sentences

Original: Bitcoins are entirely imaginary currency (*i.e.*, they have no use-value), but not particularly more so than US dollars, and could be a general currency if 300 million people similarly behaved as though it was one, *i.e.*, would do work in exchange for it.

Translation: 比特币是完全的虚拟货币（即不具备使用价值），但如果当 3 亿用户同时认为它是真实货币时（即可以用于交换），它就变得和美元一样了，可以当作一般的货币使用。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Bitcoin is real money like US dollars.
- ___ 2. Bitcoin must work relying on some central government or management.
- ___ 3. Bitcoin is virtual currency which has no central bank backing.
- ___ 4. Bitcoins can be an exchange medium and the trading using them is a lot.
- ___ 5. We can buy something like Internet service with Bitcoins.

II. Choose the best answer to each of the following questions.

1. Which of the following is wrong about the Bitcoins?
 - (A) Bitcoins are entirely imaginary currency.
 - (B) We can buy actual stuff with Bitcoins.
 - (C) Bitcoin is entirely decentralized.
 - (D) It is very easy to duplicate Bitcoins.
2. What kinds of people would Bitcoins appeal to probably?
 - (A) Soldiers
 - (B) Teachers
 - (C) Freelancers
 - (D) Accountants
3. How many years do the number of new Bitcoins created in each update that is halved take?
 - (A) 1
 - (B) 2
 - (C) 3
 - (D) 4

III. Translation

1. This provides an incentive to keep the currency running, but also attracts a lot of prospectors and speculators looking for easy money, and scammers who consider them suitably exploitable suckers.
2. These servers communicate over an Internet-based network and confirm transactions by

adding them to a ledger which is updated and archived periodically using peer-to-peer file sharing technology.

Part 3 Simulated Writing: User Guide

User guides are written to explain in layman's terms how to use software. Generally they accompany other documentation, such as system administration guides. They are generally the first part of call when something needs to be read. People read them in a hurry, usually when they are frustrated and losing patience with the software.

As the name implies, user guides are written for users to help them understand an application or system. When writing a user guide, use simple language with short sentences. This style helps the user through the application. User guides are often written for non-technical individuals. The level of content and terminology will differ considerably from for example a system administration guide, which is very detailed and complex.

Identifying Your Audience

Before you start, identify the target audience, their level of technical knowledge, and how they will use the guide. In the planning process, develop an audience definition that identifies the user, the system and the tasks.

Software is used to do specific things. Users want to know what the software can do for them, for example, how to print a page in landscape. The user guide is to teach them how the software helps them to do something.

Writing the Parts of a User Guide

The following section describes special considerations for the front matter, body, and back matter of the guide.

Front Matter

Include a cover page, table of contents and maybe a preface.

- **Cover and Title Page.** If the document is copyrighted, include a copyright notice: Copyright © 2013 The Name of Your Company. Place this on the cover (and also the title page).
- **Disclaimer.** Include a standard disclaimer inside the front cover.
- **Preface.** Use this section to reference other documents related to the software. It is common practice to include a section on "How to Use This Manual" as an introduction.
- **Contents.** Always have a table of contents, and include an index at the end of the document.

Body of the Guide

In the main body, distinguish and separate procedures from reference materials.

- **Procedures.** Procedures help the user perform specific tasks and may include:
 - When, why, and how to perform a task
 - What the screen will show after performing a task
 - Examples of tasks and program operation

Procedures involve:

1. Listing the major tasks
2. Breaking each major task into subtasks
3. Leading the user through each subtask in a series of steps
4. Using an “if-then” approach in explaining decisions that users can make

- **Learning to chunk.** Breaking tasks and information into smaller parts is called “chunking.”

In user guides, information can be divided by menu options and the consequences. Subject matter and subtasks that need to be performed divide chunks. Each chunk can form a new chapter or section of a manual. When writing procedures, number each step. If you only need a few steps, set them apart with white space. Use a consistently structured format for sections. For instance, you could

- Begin each section with an overview of the activities to perform
- Describe the input and the output
- Provide the procedures for accomplishing the tasks
- **Using the if-then approach.** When users must make choices, use an if-then approach to provide clear alternatives and consequences. And use diagrams for complicated procedures.
- **Reference materials.** Readers study reference materials when they need more information on a particular topic. Reference materials can include:
 - Program options
 - Dictionary of “hot” keys
 - Error messages
 - Troubleshooting tables
 - Commonly asked questions
 - Examples of input and output

Back Matter

Glossaries and indexes (normally placed behind the procedures) are two of the most valuable accessories in user’s guides.

- **Glossary.** This is vital when you create new, program-specific terms or when you use common terms in a special way.
 - A small glossary can appear at the front before the contents pages
 - A larger complete glossary should appear in the back matter
 - Glossary terms should be flagged the first time they appear in text
- **Index.** Any guide longer than 20 pages benefits from an index. Large documents without an index are impossible to use efficiently.

Establishing Standards

• Format

Some clients may have format preferences. Check this with the client during planning. A good format maps the organization of the guide, using clear signposts to find reference information easily.

To achieve this, use the following tactics:

- Use headings for organizing information.
- Place page numbers and section titles on every page, either in footers or headers.
- Use section numbering with restraint. As an alternative to section numbering, divide instructions into manageable segments, based on activities that the reader will perform. Provide new sections on the right-hand pages for each new division.
- Dual columns can be used effectively by putting headings in the left-hand column and the text in the right-hand column.

• Style

Use an appropriate style. Decide on the technical level of your language, how you address the user, and conventions that are required.

• Technical Language

Match the level of technical language with the audience's level of proficiency. Always underestimate the knowledge of your readers rather than overestimate it.

Limit technical terms to those the user will encounter. If you must define a large number of terms, use a glossary to supplement definitions in the text.

• Addressing the User

When writing procedures, use the active voice (e.g. Click this) and address users directly (write "you" rather than "the user"). When explaining an action, use the "command" form of the verb.

A Sample User Guide

User Guide For PST Repair Tool

PST Repair Tool is efficient in working and recovering data. It provides self- descriptive and graphical interface which is helpful for the non-technical person to understand its process easily. If it gets oversized file to recover, then it can splits data into smaller size and then easily recover it. It is also capable to recover password protected files by removing its password.

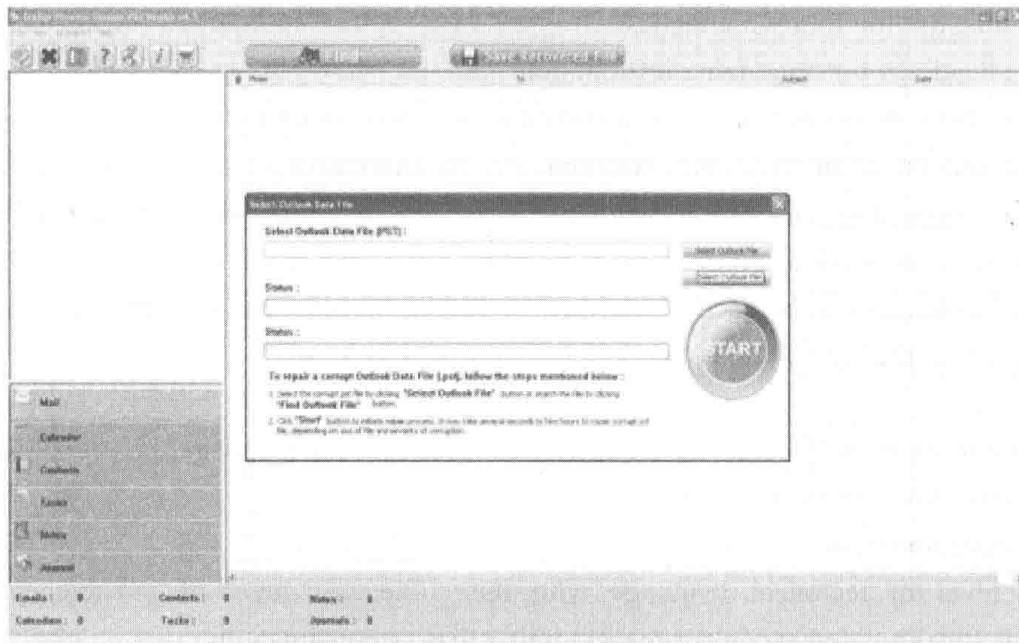
So in order to overcome from data loss situation you can use PST Repair Tool on MS Outlook. It is compatible with different versions of MS Outlook such as Outlook 98, 2000, 2003,2007, 2010. Always shut down your system properly to avoid data loss situation and keep updated PST files. If any problems occurs with your data then use PST Repair Tool to recover data and avoid the

data loss situation.

There is no need to worry because here are the steps that are needed to use the software.

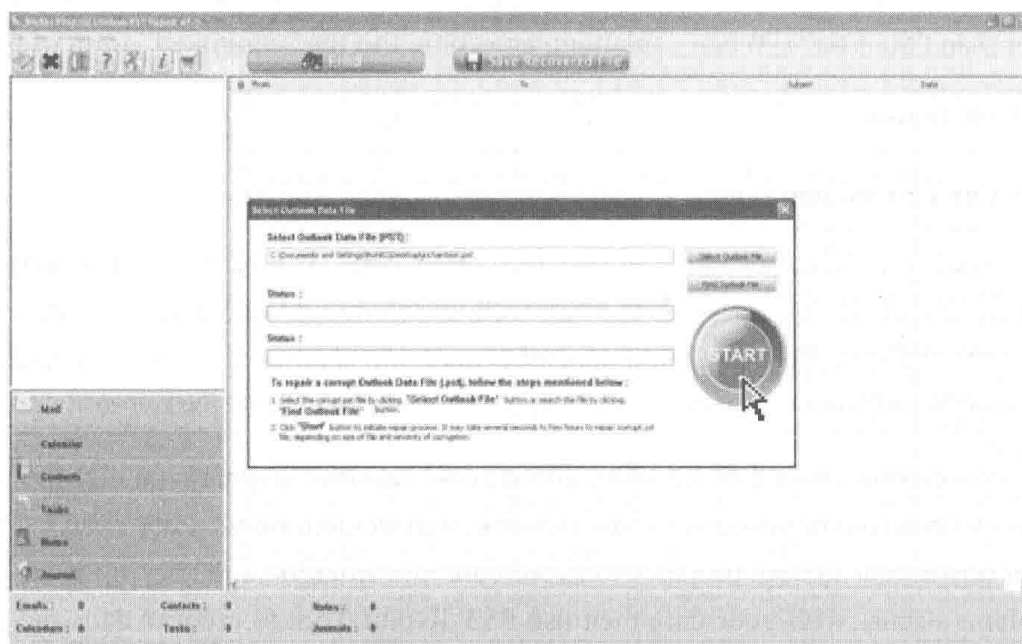
Step 1:

Firstly launch the software and hit on “Select Outlook File” in order to select the PST file that has to be recovered from its location.



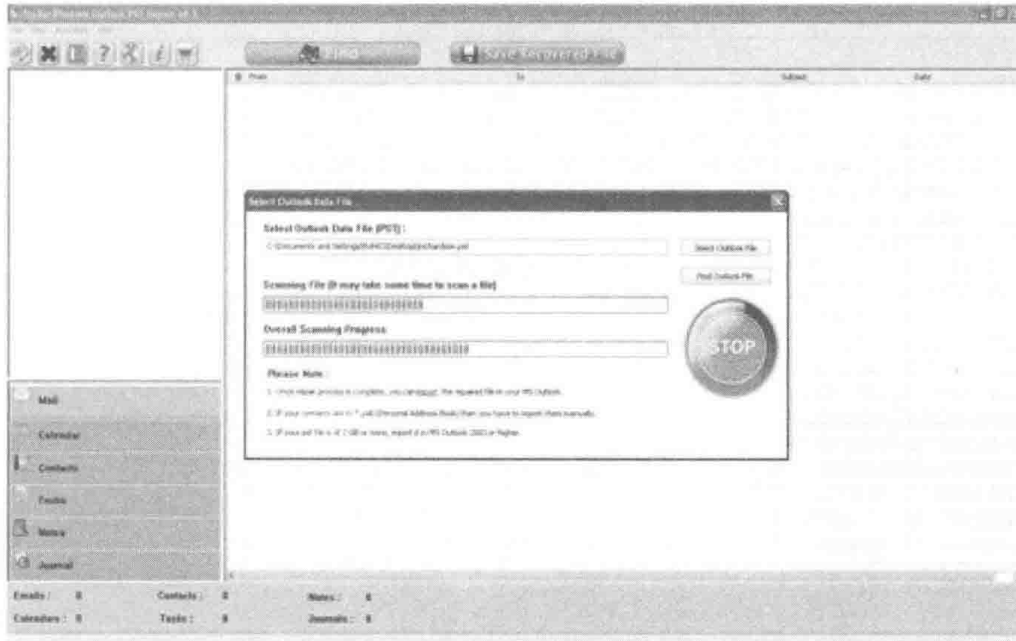
Step 2:

Now you can see the selected file part from the “Select Outlook Data File (PST)”. After that click on the “Start” button to start the scanning process.

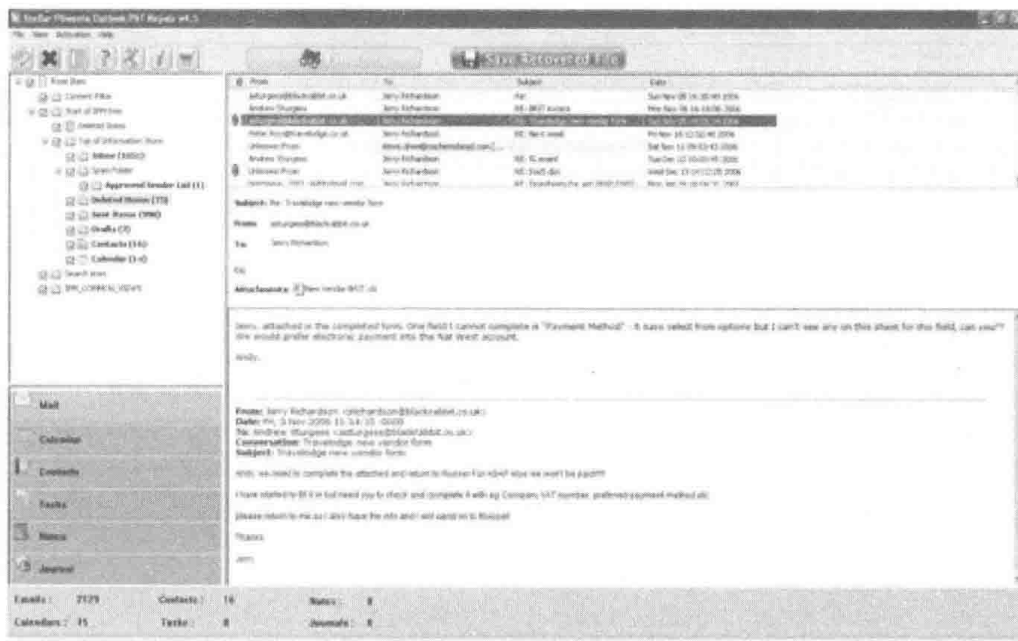


Step 3:

The Progress Bar on the screen will indicate the scanning process and it also provides the option to stop the process by clicking on the “Stop” Button.

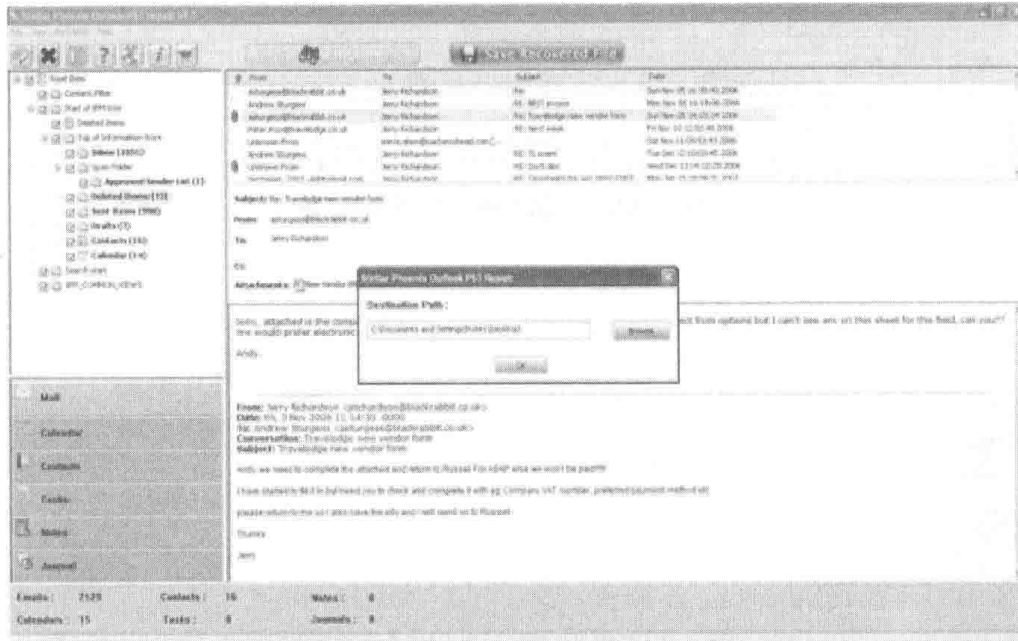
**Step 4:**

This is complete the scanning process and after that you can simply see the list of all the recovered data on the left panel of the interface. Here the software allow the user to preview all the email messages, contacts, tasks, journals, etc.

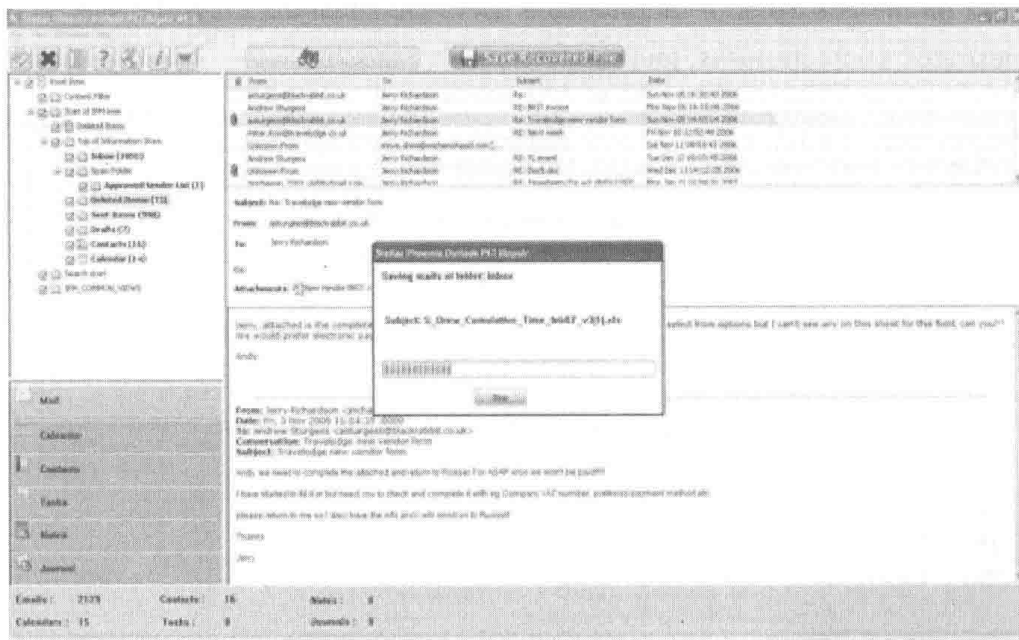


Step 5:

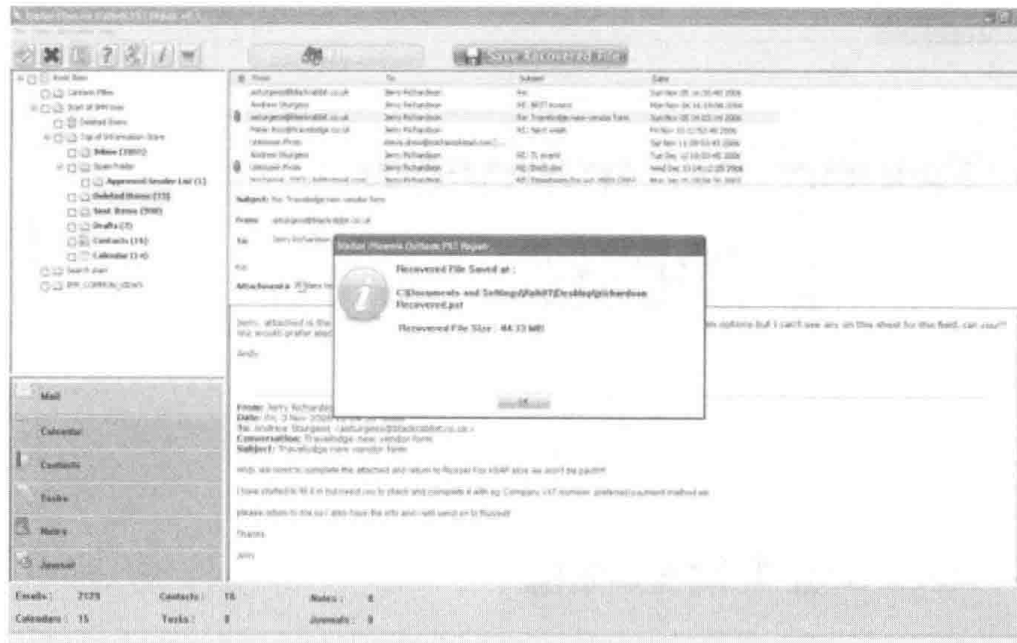
After that click on the “Save Recovered File” button to start the recovering process and here you can also specify the destination of recovered data.

**Step 6:**

The software will save all the recovered PST data at the desired location.

**Step 7:**

After saving process is completed, you can simply see a dialog box showing the “Destination Part” and the “Recovered File Size”.



Exercises

1. Is there any distinct user colless levels defined in this PST Repair Tool?
2. What should developers and system maintainers do for using or mod fying PST Repair Tool?

Taking an Interview

参加面试

Part 1 Listening & Speaking

➤ Dialogue: Interview

(Kevin's application for a position as a programmer in the technical department of Expansion has been responded by the Human Resources Department of the company. Today he comes to the company for an interview.)

Mr. Smith: Good morning, thank you for applying for our opening positions.

My name is Smith, the manager of Human Resources Department. What's your name, please?

Kevin: Good morning, Mr. Smith. My name is Kevin Li, and I am coming for the position as a programmer in the technical department.

Mr. Smith: Well, now I would like to talk with you regarding your qualifications for this interview. What can you tell me about yourself? ^[1]

Kevin: I'm an active and responsible person, and will not **give up** even if I encounter numerous **obstacles** on the way to my goal. I like to work in a team, because I believe that it is important to share the opinions with each other to get the best results. I am very interested in the latest advancements in science and technology, and have a good theoretical background and **hands-on** capabilities. I love the IT industry and want to **pursue** an IT career.

Mr. Smith: Can you talk about your education background? ^[2]

Kevin: I have just graduated from School of Software, BeiHang University in July of 2013, and obtained a bachelor degree in software engineering.

Mr. Smith: Why do you apply for a job at our company? ^[3]

Kevin: As we know, Expansion company is one of the longest established and leading software outsourcing products, services and solutions providers

[1] Replace with:

1. Tell me something about yourself.
2. How do you **value/** **judge** yourself?
3. What is your work style?
4. Why do you think that you **deserve** to get the job?

[2] Replace with:
What is your education?

[3] Replace with:

1. Why do you want to work for us?
2. What do you know about us?

in China, but also has a global **presence**. So I believe that working here can help me grow professionally. I also think I can contribute to the development of the company.

Mr. Smith: Have you ever done an **internship** that helped to prepare you for this type of work? ^[4]

Kevin: Yes. I have participated in several practical projects as a programmer in the Lab which is built by our school's cooperating with Microsoft during my undergraduate study. I participated in a hotel management information system for Four Seasons Hotel as a project manager and a management information system for the financial department of Giant company. Besides, regarding my internship, I have done a research and analysis about the global software market and completed a business report for a domestic leading IT company in English. All of the above are very valuable internship experience for me to prepare for this work, I think.

Mr. Smith: OK, that sounds great! What else do you want to know about your work?

Kevin: I want to know what other qualifications you consider necessary for this position?

Mr. Smith: Well, as I know, sometimes the **workload** of developing software may be really very heavy, so the person who is in charge of it may have to be under much pressure and must be **energetic** and capable.

Kevin: Yes, I agree with you. On this point, my internship experience has made me used to this kind of work environment.

Mr. Smith: OK, thank you for your interest in our company. If we decide to bring you onto our team, you will receive an E-mail from us within five working days.

Kevin: Thank you very much. I'm looking forward to hearing from you.

[4] Replace with:

1. How have you prepared yourself for the transition from college to the workplace?
2. Did you **get your hands on** experience in School?
3. What aspects of your abilities and experience make you think that you will succeed in this job?
4. What attracts you in this job?

Words

obstacle

['ɒbstəkl] n. 障碍, 干扰

hands-on

['hændz'ɒn] adj. 亲身实践的, 实际动手做的

pursue

[pə'sju:] v. 从事, 进行

presence

['prezns] n. 势力

internship

['intə:nʃɪp] n. 实习生职位, 实习期

workload

['wɜ:kləʊd] n. 工作量

energetic

[.enə'dʒetɪk] adj. 精力充沛的

value

['vælju:] v. 评价

judge

[dʒʌdʒ] v. 判断, 评价

deserve

[dɪ'zə:v] v. 值得, 应得

Phrases

give up

放弃

get one's hands on

把……弄到手

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other expressions on the right side.

➤ **Listening Comprehension: Expansion Company**

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- What are the core products and/or services offered by Expansion now?
 - Testing Service & Software Development
 - Software Development & Application Maintenance Service
 - Software Development & BPO Service
 - Application Maintenance & BPO Service
- Who is the largest client of Expansion among its overseas market currently?
 - Japan
 - Korea
 - Europe
 - U.S.
- Which point is not mentioned about the future actions of Expansion according to the article?
 - Increase earning and profit
 - Develop strong management team
 - Economize physical resources
 - Expand new markets

Words**headquarter**

[ˌhed'kwɔ:tə] v. 总部设于……, 总公司
设于……

outsourcing

['aut,sɔ:sɪŋ] n. 外购, 外包

Words

rich

[ritʃ] adj. 宝贵的

pool

[pu:l] n. 资源的集合, 联营

margin

['mɑ:dʒɪn] n. 利润

advance

[əd'vɑ:ns] v. 提升, 提高

Abbreviations

BPO

Business Process Outsourcing

业务过程外包

➤ Dictation: Software Configuration Management

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

When you build computer software, ____1____ happens. And because it happens, you need to control it ____2____. Software configuration management, **SCM** for short, is a set of activities that are designed to ____3____ change by identifying the work products that are ____4____ to change, ____5____ relationships among them, defining mechanisms for managing different ____6____ of these work products, controlling changes that are **imposed**, and auditing and ____7____ on the changes that are made. Another name for it is “change control.”

In his book *Code Complete*, Steve McConnell **depicts** that: “If you don’t control changes to requirements, you can **end up** writing code for parts of the system that are ____8____ eliminated. You can write code that’s incompatible with new parts of the system. You might not ____9____ many of the incompatibilities until ____10____ time, which will become **finger-pointing** time because nobody will really know what’s going on.”

Since programming projects are so volatile today, SCM is especially useful to programmers. It focuses on a program’s ____11____, source code, ____12____, and test data.

SCM systems are based on a simple ____13____: the definitive copies of your files are ____14____ in a central repository. People ____15____ copies of files from the repository, work on those ____16____, and then check them ____17____ in when they are finished. SCM systems manage and track ____18____ by multiple people against a single **master** set. All SCM systems provide the following essential ____19____: **concurrency** management, ____20____ and **synchronization**.

Words**impose**

[im'pəuz] v. 强制实行

depict

[di'pikt] v. 描述, 描写

finger-pointing

['fɪŋgə'pɔɪntɪŋ] n. 指责, 责难

master

['mɑ:stə] adj. 支配的, 总的, 最重要的

concurrency

[kən'kʌrənsi] n. 并行

synchronization

[ˌsɪŋkrənaɪ'zeɪʃən] n. 同步

Phrases

end up

结束

Abbreviations

SCM

Software Configuration Management

软件配置管理

Part 2 Reading and Translation**➤ Section A: Careers for Computer Professionals**

In 1999, the U.S. Congress crafted an amendment to the Fair Labor Standards Act that essentially defines a computer professional as any person whose primary occupation involves the design, configuration, analysis, development, modification, testing, or security of computer hardware or software. [1]

Many computer professionals work in an Information System department—the **wing** of a business or organization responsible for computer, data, software, and support services. An **IS** department is also responsible for prioritizing an organization's information needs, modifying old systems as necessary, and creating new systems. Most IS department are **headed** by a chief information officer (CIO), or director. The CIO heads a hierarchy of computer professionals, who might be organized as in Figure 1.

The following descriptions highlight typical responsibilities and skills for various IS department jobs: A systems analyst investigates the requirements of a business or organization, its employees, and its customers in order to plan and implement new or improved computer services. This job requires the ability to identify problems and research technical solutions. Good communications skills are essential for interacting with managers and other employees.

A computer programmer (sometimes described as a programmer/analyst) designs, codes, and tests computer programs. In addition, programmers modify existing programs to **meet** new

requirements or eliminate bugs. Computer programming requires concentration and a good memory for the countless details that **pertain** to a programming project.

A security specialist analyses a computer system's **vulnerability** to threats from viruses, worms, unauthorized access, and physical damage. Security specialists install and configure firewalls and antivirus software. They also work with **management** and employees to develop policies and procedures to protect computer equipment and data. A security specialist must have wide-ranging knowledge of computers as well as communications protocols that can be applied for a quick resolution to any crisis that occurs.

A database administrator analyzes a company's data to determine the most effective way to collect and store it. Database administrators create databases, data entry forms, and reports. They also define backup procedures, provide access to authorized users, and **supervise** the day-to-day use of databases.

A network specialist/administrator plans, installs, and maintains one or more local area networks. These specialists also provide network accounts and access rights to **approved** users. They troubleshoot connectivity problems and respond to requests from the network's users for new software. Network specialists/administrators might be responsible for maintaining the security of a network, **plus** they often **pick up** Webmaster duties to maintain an organization's Web site.

A computer operator typically works with **minicomputers**, mainframes, and supercomputers. Computer operators monitor computer performance, install software **patches** and **upgrades**, perform backups and restore data as necessary.

A technical support specialist troubleshoots hardware and software problems. Good **interpersonal** skills and patience are required for this job.

A Web site designer creates, tests, **posts**, and modifies Web pages. A good sense of design and artistic talent are required for this job, along with an understanding of how people use graphical user interfaces. Familiarity with Web tools, such as HTML, XML, JavaScript, and ActiveX, is becoming more important for this job, **as** is knowledge of computer programming and database management.

A technical writer creates documentation for large programming projects and writes the **online** or printed user manuals that accompany computers, peripheral devices, and software. Some

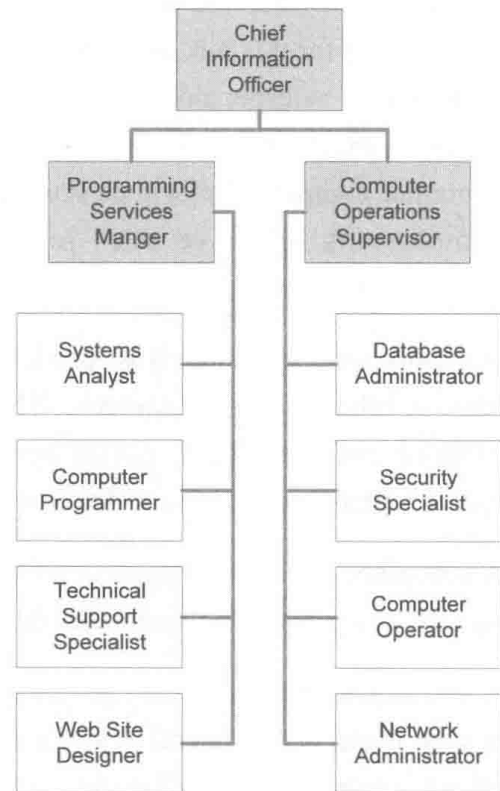


Figure 1 This organizational structure might typically be in a mid-size business

technical writers work for computer magazines, writing **columns** about the latest hardware products, software, and automated business solutions. Good writing and communications skills are valuable for this job, as is an ability to quickly learn how to use new computers and software. A computer salesperson, or sales **rep**, sells computers or IT services. Sales reps might **pay** personal **visits to** potential corporate customers or work at the **order** desk of a mail-order computer company. Sales reps' starting salaries tend to be low but are usually **supplemented** by **commissions**. Effective sales reps have good interpersonal skills, an ability to remember technical specifications, and an understanding of business problems and solutions.

A quality assurance specialist participates in alpha and beta test cycles of software, looking for bugs or other usability problems. This job title sometimes refers to assembly-line workers who examine and test **chips**, circuit boards, computers, and peripheral devices. An effective **QA** specialist has a good eye for detail and a passion for perfection.

A computer engineer designs and tests new hardware products, such as computer chips, circuit boards, computers, and peripheral devices.

A manufacturing technician participates in the **fabrication** of computer chips, circuit boards, system units, or peripheral devices. Some of these jobs require basic **screwdriver** skills, whereas others require special training in **microlithography**.

The Information Technology Association of America (ITAA) categorizes IT jobs into eight clusters. Sample job titles are listed here:

Database Administration and Development: Database Administrator, Database Analyst, Database Developer, Database Manager, Database Security Expert, **DSS** (Decision Support Services), and Knowledge Architect.

Digital Media: 2-D/3-D **Artist**, **Animator**, Audio/Video Engineer, Designer, Media Specialist, Media/Instructional Designer, Multimedia Author, Multimedia Authoring Specialist, Multimedia Developer, Multimedia Specialist, Producer, Streaming Media Specialist, and Virtual Reality Specialist. Enterprise System Analysis and Integration: Application **Integrator**, Business Continuity Analyst, Cross-Enterprise Integrator, Data Systems Designer, Data Systems Manager, **E-business** Specialist, Information Systems **Architect**, Information Systems Planner, Systems Analyst, Systems Architect, and Systems Integrator.

Network Design and Administration: Communications Analyst, Network Administrator, Network Analyst, Network Architect, Network Engineer, Network Manager, Network Operations Analyst, Network Security Analyst, Network Specialist, and Network Technician.

Programming/Software Engineering: Applications Analyst, Applications Engineer, Business Analyst, Computer Engineer, OS Designer/Engineer, OS Programmer/Analyst, Program Manager, Programmer, Programmer/Analyst, Software Applications Specialist, Software Architect, Software Design Engineer,

Software Engineer, Software QA Specialist, and Software Tester.

Technical Support: **Call Center** Support Rep, Customer Liaison, Customer Service Rep, Customer Support Professional, **Help Desk** Specialist, Help Desk Technician, PC Support Specialist, PC Systems Coordinator, Product Support Engineer, Sales Support Technician, Technical Account Manager, Technical Support Engineer, and Technical Support Representative.

Technical Writing: Desktop Publisher, Document Specialist, Documentation Specialist, Editor, Electronic Publications Specialist, Electronic Publisher, Instructional Designer, Online Publisher, Technical Communicator, Technical Editor, Technical Publications Manager, Technical Writer.

Web Development and Administration: Web Administrator, Web Architect, Web Designer, Web Page Developer, Web Site Developer, Web Specialist, and Webmaster.

Words

Congress

[ˈkɒŋɡres] n. (美国)国会, 议会

craft

[kra:ft] v. 精心策划, 制作

amendment

[əˈmendmənt] n. 修正案, 修正条款

wing

[wiŋ] n. 部门, 成员组织, 附属机构

head

[hed] v. 负责, 领导

meet

[mi:t] v. 满足(需求、要求等)

pertain

[pəˈtein] v. 关于, 有关

vulnerability

[ˌvʌlnərəˈbiləti] n. 弱点

management

[ˈmænidʒmənt] n. 管理人员, 管理部门

supervise

[ˈsju:pəvaiz] v. 监督, 管理

approve

[əˈpru:v] v. 批准, 通过

plus

[plʌs] conj. 并且, 而且

Webmaster

[ˈwebˈmɑ:stə] n. 网站维护者, 网站管理员

minicomputer

[ˈminikəm.pju:tə] n. 小型计算机

patch

[pætʃ] n. 补丁(程序)

upgrade

[ˈʌpgreɪd] n. (尤指计算机设备等的)升级

interpersonal

[ˌɪntəˈpɜ:sənəl] adj. 人际(关系)的

post

[pəʊst] v. 公布, 发布

as

[æs] pron. 同样

online

[ˈɔ:n,laɪn] adj. 联机的, 在线的

column

[ˈkɒləm] n. 专栏(文章)

order

[ˈɔ:də] n. 订购, 订单

supplement

[ˈsʌplɪmənt] v. 补充

commission

[kəˈmɪʃən] n. 佣金, 回扣, 代办费, 手续费

Words

chip

[tʃip] n. 芯片

fabrication

[ˌfæbrɪˈkeɪʃən] n. 制造, 生产

screwdriver

[ˈskruːdraɪvə] n. 螺丝刀

lithography

[liˈθɒɡrəfi] n. 平版印刷术

artist

[ˈɑːtɪst] n. (某方面的)能手, 大师, 画家, 艺术家

animator

[ˈænɪmeɪtə] n. 漫画家, 动画片绘制者

integrator

[ˈɪntɪɡreɪtə] n. 综合者, 合成者

architect

[ˈɑːkɪtekt] n. 架构师, 设计师

liaison

[liˈeɪzɔːn] n. 联络员

account

[əˈkaʊnt] n. 客户

communicator

[kəˈmjuːnɪkeɪtə] n. 传播者, 通信员

Phrases

pick up

(尤指偶然地、无意地、容易地)获得

pay a visit to

访问, 参观

call center

呼叫中心

help desk

(向网络用户提供信息和技术支持的)用户服务部

Abbreviations

IS	Information System	信息系统
CIO	Chief Information Officer	首席信息官
rep.	representative	代理人, 代表
QA	Quality Assurance	质量保证
ITAA	Information Technology Association of America	美国信息技术协会
DSS	Decision Support Services	决策支持服务
E-business	Electronic Business	电子业务
PC	Personal Computer	个人计算机

Complex Sentences

Original: In 1999, the U.S. Congress crafted an amendment to the Fair Labor Standards Act that essentially defines a computer professional as any person whose primary occupation involves the design, configuration, analysis, development, modification, testing, or security of computer hardware or software.

Translation: 1999 年, 美国国会对 (美国) 公平劳动标准法做了修正, 从本质上将计算机专业人员定义为, 其主要职业涉及计算机硬件或软件的设计、配置、分析、开发、修改、测试或安全的任何人。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. The scope of the computer professionals' work covers both hardware and software.
- ___ 2. CIO is the leader of IS department in a business or organization in general.
- ___ 3. It is unnecessary for Web Site Designers to know the knowledge of computer programming or database management.
- ___ 4. Technical writers may work either for IT companies or for computer magazines.
- ___ 5. Database administrators work with management and employees to develop policies and procedures to protect computer equipment and data.

II. Choose the best answer to each of the following questions.

- 1. Which statement is right about the careers for computer professionals?
 - (A) All the computer professionals should belong to Information System department of a business or organization.
 - (B) Good communications skills are essential for various computer professionals.
 - (C) Business analyst should be organized under the Computer Operations Supervisor in Figure 9-1.
 - (D) Applications Engineer is a job title which belongs to the cluster of Enterprise System Analysis and Integration categorized by ITAA.
- 2. Which of the following is right about the responsibilities and skills of various IS department jobs?
 - (A) Security specialist develops and configures firewalls and antivirus software.
 - (B) Quality assurance specialist is responsible for modifying existing programs to eliminate bugs.
 - (C) Computer salesperson requires the ability to identify problems and research technical solutions.
 - (D) Webmaster's duty is to maintain an organization's Web site.
- 3. Assume that you are a software engineer in a company, which of the following situations is likely wrong according to this article?
 - (A) If you want to have access to the local area network in your business, you should appeal to the database administrator.
 - (B) If your computer can not work, you should refer yourself to the technical support engineer.
 - (C) If you want an online user guide for your software, you should call the technical writer.
 - (D) If you have something confused on requirements, you should seek helps from the systems analyst.

III. Translation

1. Computer programming requires concentration and a good memory for the countless details that pertain to a programming project.
2. A security specialist must have wide-ranging knowledge of computers as well as communications protocols that can be applied for a quick resolution to any crisis that occurs.

➤ Section B: 3D Printing 三维打印

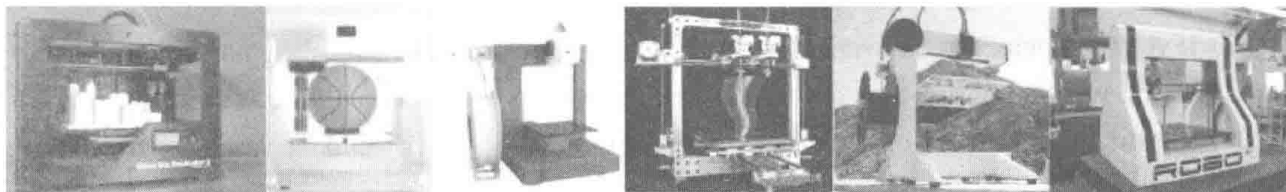
You've heard of 3D printing from newscasters and journalists, astonished at what they've witnessed. A machine **reminiscent of** the **Star Trek Replicator**, something magical that can create objects out of thin air. It can "print" in plastic, metal, nylon, and over a hundred other materials. It can be used for making **nonsensical** little models like the over-printed **Yoda**, yet it can also print manufacturing prototypes, end user products, **quasi-legal** guns, aircraft engine parts and even human organs using a person's own cells.

We live in an age that is witness to what many are calling the Third Industrial Revolution. 3D printing, more professionally called **additive** manufacturing, moves us away from the Henry Ford era mass production line, and will bring us to a new reality of customizable, **one-off** production.

Need a part for your washing machine? As it is now, you'd order from your repairman who gets it from a distributor, who got it shipped from China, where they mass-produced thousands of them at once, probably **injection-molded** from a very expensive mold. [1] In the future, the beginning of which is already here now, you will simply 3D print the part right in your home, from a CAD file you downloaded. If you don't have the right printer, just print it at your local **fab**.

3D printers use a variety of very different types of additive manufacturing technologies, but they all share one core thing in common: they create a three dimensional object by building it layer by successive layer, until the entire object is complete. It's much like printing in two dimensions on a sheet of paper, but with an added third dimension: UP, the Z-axis.

Each of these printed layers is a thinly-sliced, horizontal cross-section of the eventual object. Imagine a multi-layer cake, with the baker laying down each layer one at a time until the entire cake is formed. 3D printing is somewhat similar, but just a bit more precise than 3D baking.



While most people have yet to even hear the term 3D printing, the process has been in use for decades. Manufacturers have long used the printers in the design process to create prototypes for traditional manufacturing. But until the last few years, the equipment has been expensive and slow.

Now, fast 3D printers can be had for tens of thousands of dollars, and end up saving the companies many times that amount in the prototyping process. For example, Nike uses 3D printers to create multi-colored prototypes of shoes. They used to spend thousands of dollars on a prototype and wait weeks for it. Now, the cost is only in the hundreds of dollars, and changes can be made instantly on the computer and the prototype reprinted on the same day.

Some companies are using 3D printers for **short run** or custom manufacturing, where the printed objects are not prototypes, but the actual end user product. As the speeds of 3D printing go up and the prices come down, look for more and more of this. And expect more availability of personally customized products.

Even if you don't design your own 3D model, you can still print some very cool pieces. There are model **repositories** such as Thingiverse, 3D Parts Database, and 3D Warehouse that have model files you can download for free.

What do all these people print? It's limitless. Some print things like jewelry, some print replacement parts for appliances such as their dishwasher, some invent all sorts of original things, some create art, and some make toys for their kids. With the many types of metal, plastic, glass, and other materials available (even gold and silver), just about anything can be printed.

This is a **disruptive** technology of **mammoth proportions**, with effects on energy use, waste, customization, product availability, art, medicine, construction, the sciences, and of course manufacturing. It will change the world as we know it.

Words

newscaster

['nju:z.kɑ:stə] n. (电台或电视的)新闻播音员

nonsensical

['nɒn'sensɪkl] adj. 无意义的, 荒谬的, 愚蠢的

additive

['ædətɪv] adj. 附加的

one-off

['wʌnɒf] adj. 一次性的

fab

[fæb] n. 代工厂

repository

[ri'pɒzə'tɔ:ri] n. 仓库

disruptive

[dis'rʌptɪv] adj. 颠覆性的, 破坏的

mammoth

['mæməθ] adj. 庞大的, 巨大的

proportion

[prəu'pɔ:ʃən] n. 部分, 比例

Phrases

reminiscent of	表明……的, 暗示……的
Star Trek Replicator	星际迷航的复制
Yoda	尤达(《星球大战》里重要的人物)
quasi-legal	准法律的, 准合法的
injection-mold	注塑模具
short run	短期(指企业无法改变生产规模的较短期限)

Complex Sentences

Original: As it is now, you'd order from your repairman who gets it from a distributor, who got it shipped from China, where they mass-produced thousands of them at once, probably injection-molded from a very expensive mold.

Translation: 在这个时代, 你所订购的零件或许来自于一个用非常昂贵的模具注塑而成的一起批量生产的成千个零件, 通过船运从中国运至分销商, 分销商再交给修理工, 最后到达用户的手中。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. 3D printers are the same as the 2D printers.
- ___ 2. 3D printers create a three dimensional object by building it layer by successive layer, until the entire object is complete.
- ___ 3. 3D printing is much like printing in two dimensions on a sheet of paper, but with an added Z-axis.
- ___ 4. In the 3D world, a 3D printer also needs to have instructions for what to print.
- ___ 5. The 3D modeling program that creates a Computer Aided Design (CAD) file can be sent to the 3D printer.

II. Choose the best answer to each of the following questions.

1. Which of the following is wrong about 3D printers?
 - (A) It's much like printing in two dimensions on a sheet of paper, no others.
 - (B) Even if you don't design your own 3D model, you can still print some very cool pieces.
 - (C) If you don't have a 3D printer, you can print something at your local fab.
 - (D) As the speeds of 3D printing go up and the prices come down, more and more people can afford to use 3D printers.
2. Which of the following can be printed by 2D printers?
 - (A) Football

- (B) Pen
 - (C) Table
 - (D) Photo
3. Which word is appropriate for describing the technology of 3D printing?
- (A) Old
 - (B) Disruptive
 - (C) Futureless
 - (D) Obsoleted

III. Translation

This is a disruptive technology of mammoth proportions, with effects on energy use, waste, customization, product availability, art, medicine, construction, the sciences, and of course manufacturing.

Part 3 Simulated Writing: Resume

A resume is the key job search tool and should be limited to one page or two pages if you have substantial experience. Because the resume gives potential employer's a first impression of you, take the time to make sure that it is well organized, well designed, easy to read, and free of errors. Only use high-grade top quality paper.

Analyzing Your Background

Determine what kind of job you are seeking and what information about you and your background would be most important and related to that kind of job. List the following:

- Schools attended, degrees, major field of study, academic honors, grade point average (if recently graduated), particular and relevant academic projects.
- Jobs held, primary and secondary duties in each of them, when and how long you held each job, promotions, skills you developed in your jobs.
- Other experiences and skills you have developed that would be of value in the kind of job you are seeking; extracurricular activities that have contributed to your learning experience; leadership, interpersonal, and communication skills you have developed; any collaborative work you have performed; computer skills you have acquired.

Organizing Your Resume

Most resumes have information arranged chronologically in the following categories:

- Heading (name and contact information)
- Job Objective (optional)
- Education

- Employment Experience
- Computer Skills (optional)
- Honors and Activities
- References

If you are recent graduate without much work experience, list education first. If you have many years of job experience, including jobs directly related to the kind of position you are seeking, list employment experience first. In your education and employment sections, use a chronological sequence with most recent experience first, the next most recent experience second, and so on.

- **Heading**

At the top of your resume, include all of your contact information including: name, address, telephone, fax, and E-mail address. Make sure that your name stands out on the page. If you have just one address, center it at the top of the page. If you have two addresses such as school and home or temporary and permanent, place school or temporary address on the left side of the page and your permanent or home address on the right side. Place both addresses underneath your name.

- **Objective**

A job objective introduces the material in a resume and helps the reader quickly understand your goal. Write your objective in three lines or less. For example: a full-time software programmer/analyst aimed at solving engineering problems.

- **Education**

List the school(s) you have attended, the degrees you received and the dates you received them, your major field(s) of study, and any academic honor you have earned. Include your grade point average only if it is 3.0 or higher. List courses only if they are unusually impressive or if your resume is otherwise sparse.

- **Employment Experience**

You can organize your employment experience chronologically, starting with your most recent job and working backward under a single major heading called "Experience", or you could also organize your experience functionally by clustering similar types of jobs into one or several sections with specific headings such as "Management Experience" or "Major Accomplishments".

- Include jobs or internships when they relate directly to the position you are seeking.
- Include extracurricular experiences, especially those involving a leadership position or community-service.
- Under each job or experience, provide a concise description of your primary and secondary responsibilities.
- Use action verbs with precision and conciseness. Do not use "I".

- **Computer Skills**

If you are applying for a job that requires computer knowledge, include specific languages, software, and hardware you are familiar with.

- **Honors and Activities**

If there is still room on the resume (less than 2 pages) items such as fluency in foreign languages, writing and editing abilities, specialized technical knowledge, student or community activities, professional or club memberships, and published works. Be selective: do not duplicate information given in other categories and include only information that supports your employment objective.

- **References**

Unless your resume is sparse, avoid listing references. For an interview, have a printed list of references available and bring them with you. Your list should include the main heading "References for [your name]".

A Sample Resume

Kevin Li

School of Software

Beihang University, Beijing 100191, P.R.China

kevin_li@buaa.edu.com

0086-13911234567

Objective

To obtain a challenging position as a software engineer with an emphasis in software design and development.

Education

2009.9-2013.7 School of Software, Beihang University.

Research and Work Experience

Research Assistant in Digital Beijing Project “Intelligent Traffic Management System”, 5/2012 - 5/2013.

Software development of Management Information System for the financial department of Giant Company, 10/2011 - 4/2012.

Research and analysis in the global software market and finally completed a business report for a domestic leading IT company, 4/2011 - 8/2011.

Teaching Assistant for C++ Programming in School of Software, Beihang University, 9/2010 - 1/2011.

Main Academic Courses

Advanced

Mathematics

Data Structure

C++ Object-Oriented Programming

Conspectus of Database Management System

Computer Network Principle

Software Engineering

Principles and Techniques of Compilers

Operating Systems

Business Communication and Technical Writing

Computer Skills

Skillful with

- Programming: Python, C, C++, C#, Java
- Database: SQL Server, Oracle

- Web Design: HTML, XML, JavaScript, JSP, ASP.NET
- Operating Systems: Linux, Windows

English Abilities

Have a good command of both spoken and written English.

Exercises

There are two scenarios below, please choose one and write a resume.

Scenario 1: As an undergraduate in grade 4, you want to apply an internship position in the R&D center, a financial company.

Scenario 2: With 2 years event planning experience, you want to join in the University Student Union.

Beginning Your Work 开始工作

Part 1 Listening & Speaking

► Dialogue: Beginning Your Work

(Congratulations on Kevin becoming a new employee as a programmer in the Information System Department of Expansion Company! Today is his first day for work. There will be some new colleagues and fresh working environment coming along with him.)

Kevin: Nice to meet you, Mr. John. My name is Kevin Li, and I'd like to report for work.

Mr. John (the CTO of Expansion): Welcome to our company, Kevin. You'll work in the Information System Department. Ok, now please let me introduce our colleagues to you.

(Kevin comes into the office following Mr. John.)

Mr. John: Hello, everybody. Today I'd like to introduce a new colleague to you. His name is Kevin Li and he will work with you from today.

Kevin: How do you do?

A, B, C: How do you do? Mr. Li, welcome!

Kevin: Thanks! I have just graduated. This is my first time working in a multinational company. I hope you can help me **adapt** from school life **to** working in a professional environment.

A: No problem, Mr. Li. I'm Jenny, in charge of ^[1] the office work. This is your desk.

Kevin: Thank you very much, Ms. Jenny.

A: You call me Jenny. Everybody in our company is called by his or her first name. It's been our tradition since the company was established.

[1] Replace with:

1. take charge of
2. be responsible for

Kevin: OK, I'll try.

B&C: Very glad to know you. We believe we will be able to cooperate well in the future.

Kevin: So do I.

Mr. John: Kevin, if you have any questions please feel free. The people here are all very kind. Any other questions?

Kevin: All right. **By the way**, excuse me. Would you please tell me what my specific tasks and responsibilities are? ^[2]

Mr. John: You'll spend most of this week to get familiar with the work in the office and read some documents about a project which you will participate in from the next week. Do you have any other questions?

Kevin: **Not yet**, but I'm sure I will have more questions later. Thanks.

(It is time for **ringing out** after have been working for a whole day.)

Jenny: Kevin, you've been working for nearly one day. How do you feel about the job? ^[3]

Kevin: Not bad. Thank you for your help. I've been busy all day and feeling a bit tired.

Jenny: I had the same feeling when I first came to work here. But after a period of time, I felt much better. I'm sure you'll **get used to** the job.

Kevin: I also feel that the work efficiency is very high and it is also challenging. Everybody here works hard and they are cooperative. And you all have strong capabilities and professional skills. I really like such a working environment.

[2] Replaced with:

Do you mind if I ask you something about my work responsibilities in the company?

[3] Replace with:

1. How are you feeling about the job?
2. What do you think about the job?

Phrases

adapt oneself to	使适应
by the way	顺便
not yet	还没有
ring out	(根据考勤钟) 打卡下班
get used to	习惯

Abbreviations

CTO

Chief Technology Officer

首席技术官

Exercises

Work in pairs, and make up a similar conversation by replacing the statements with other

expressions on the right side.

➤ Listening Comprehension: The Organizational Structure of a Company

Listen to the article and the following 3 questions based on it. After you hear a question, there will be a break of 10 seconds. During the break, you will decide which one is the best answer among the four choices marked (A), (B), (C) and (D).

Questions:

- How many major leading positions in most companies are mentioned in the article?
(A) Four
(B) Five
(C) Six
(D) Seven
- Whom should you be directly under the supervision of if you are a software engineer in an IT company?
(A) CEO
(B) CTO
(C) COO
(D) CSO
- Which task does not likely belong to the responsibilities of the CFO according to the article?
(A) Recruiting a staff member
(B) Analyzing the margins changing trend
(C) Taking revenue forecast
(D) Making a financial planning

Words

chief

[tʃi:f] n. 长官, 首领

board

[bɔ:d] n. 理事会, 董事会

oversee

['əʊvə'si:] v. 管理, 监督

acronym

['ækronim] n. 只取首字母的缩写词

rank

[ræŋk] n. 等级

Phrases

take the lead

为首, 带头

under the supervision of

在……的监督下, 在……的指导下

Abbreviations

CEO	Chief Executive Officer	首席执行官, 执行总裁
CFO	Chief Financial Officer	首席财务官, 财务总监
COO	Chief Operating Officer	首席运营官
CBDMO	Chief Business Development and Marketing Officer	首席商业扩展和营销官
CSO	Chief Strategy Officer	首席策略官

➤ Dictation: Open Source Movement

This article will be played three times. Listen carefully, and fill in the blanks with the words you have heard.

In the early days of computing, software came _____ 1 _____ with the computer, including the source code for the software. Programmers **adjusted** and **adapted** the programs and happily shared the improvements they made. In the 1970s, firms began _____ 2 _____ the source code, and software became big _____ 3 _____.

As opposed to being kept _____ 4 _____, "open source" means that code is freely _____ 5 _____ by those who write it, which _____ 6 _____ the power of distributed peer review and transparency of process. The promise of open source is better quality, higher **reliability**, more **flexibility**, lower cost, and an end to **predatory** vendor _____ 7 _____.

In 1960's, researchers with access to the Advanced Research Projects Agency Network (ARPANET) used a process called Request for Comments to develop telecommunication network _____ 8 _____. _____ 9 _____ by **contemporary** open source work, this collaborative process **led to** the _____ 10 _____ of the Internet in 1969. The decision in the free software movement to use the label "open source" **came out of** a strategy _____ 11 _____ held at Palo Alto, California, in _____ 12 _____ to Netscape's January 1998 _____ 13 _____ of a source code release for Navigator.

So far, much of the _____ 14 _____ programming that **powers** the Internet, creates _____ 15 _____, and produces software is the result of "open source" code. Leaving source code open has _____ 16 _____ some of the most sophisticated developments in computer technology, including, most notably, Linux and Apache, which **pose** a significant _____ 17 _____ to Microsoft in the marketplace.

In his book The Success of Open Source, Steven Weber argues that the success of open source is not a **freakish** _____ 18 _____ to economic principles. The open source community is guided by standards, rules, decision-making _____ 19 _____, and **sanctioning** _____ 20 _____.

Words**adjust**

[ə'dʒʌst] v. 调节, 调整

adapt

[ə'dæpt] v. 使适合, 使适应

reliability

[ri,laɪə'biliti] n. 可靠性

flexibility

[fleksə'biliti] n. 弹性, 适应性, 灵活性

predatory

['predətəri] adj. 掠夺性的, 损人利己的

contemporary

[kən'tempərəri] adj. 属于同时期的, 同时代的

power

['paʊə] v. 促进, 推动

pose

[pəʊz] v. 引起, 造成

freakish

['fri:kiʃ] adj. 反常的, 奇怪的

sanction

['sæŋkʃən] n. & v. 批准, 认可, 同意

Phrases

lead to

导致

come out of

出自

Abbreviations

ARPANET

Advanced Research Projects Agency
Network美国国防部高级研究计划
管理局网络**Part 2 Reading and Translating****➤ Section A: The Organizational Structure of a Company**

Consider any **sizable** organization with which you are familiar. Its purpose is to perform a service or deliver a product. If it's nonprofit, for example, it may deliver the service of educating students or the product of food for **famine** victims. If it's profit-oriented, it may, for example, sell the service of fixing computers or the product of computers themselves. Information—whether computer-based or not—has to flow within an organization in a way that will help managers, and the organization, achieve their goals. [1] To this **end**, organizations are often structured horizontally and vertically—horizontally to reflect functions and vertically to reflect management levels.

Depending on the services or products they provide, most organizations have departments that perform six functions: research and development (R&D), production (or operations), marketing and sales, accounting and finance, human resources (personnel) and information systems (IS). (See Figure 1)

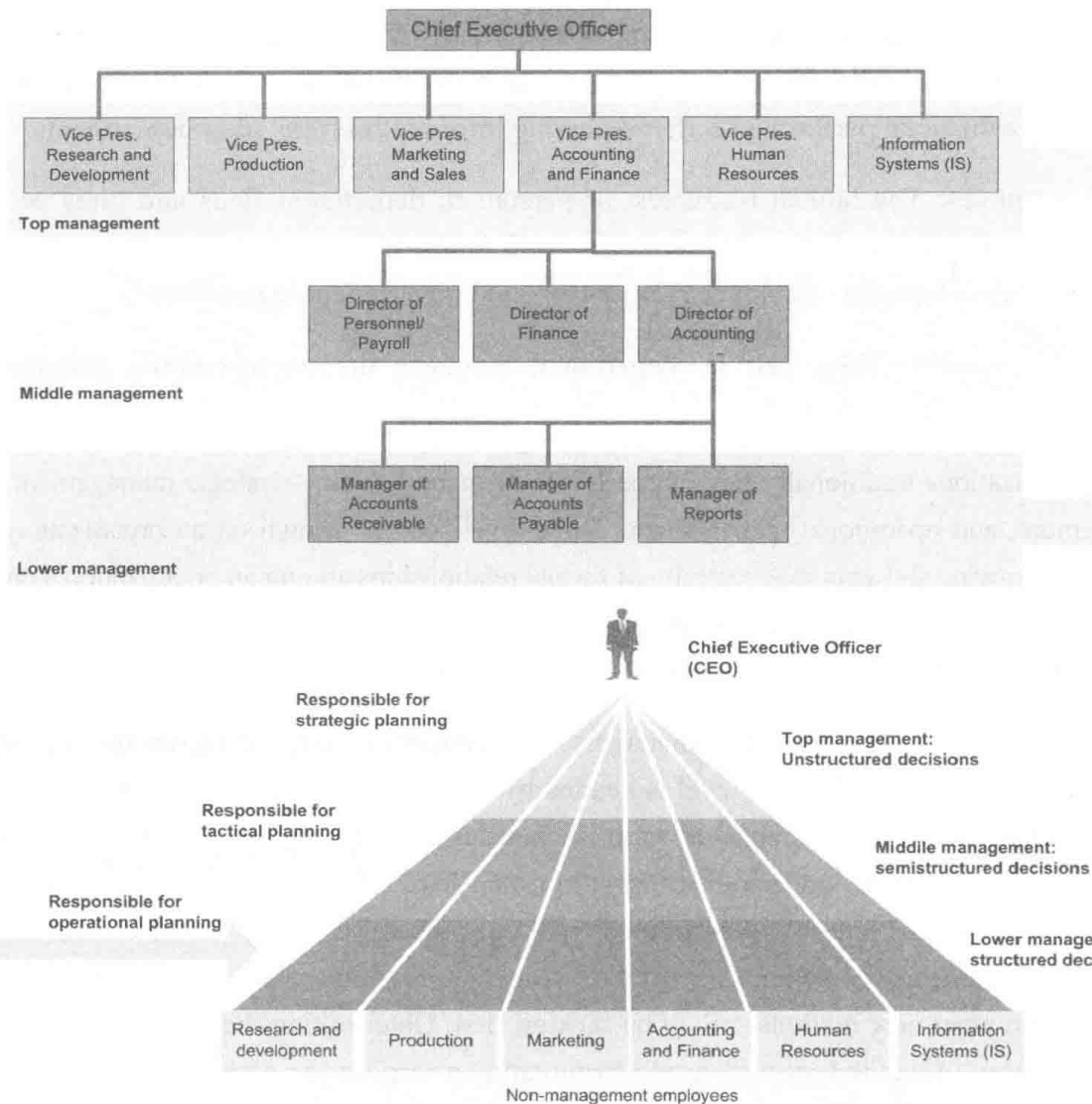


Figure 1 Organization chart

Research and development: The research and development (R&D) department does two things:

- (1) It conducts basic research, relating discoveries to the organization's current or new products.
- (2) It does product development and tests and modifies new products or services created by researchers.

Production (operations): The production department makes the product or provides the service. In a manufacturing company, it takes the raw materials and has people or machinery turn them into finished goods. In another type of company, this department might manage the purchasing, handle the inventories, and control the flow of goods and services.

Marketing and sales: The marketing department oversees advertising, **promotion**, and sales. The people in this department plan, **price**, advertise, promote, **package**, and distribute the services or goods to customers or clients.

Accounting and finance: The accounting and finance department handles all financial matters. It

handles cash management, pays bills and taxes, issues **paychecks**, records payments, makes investments, and compiles financial statements and reports. It also produces financial budgets and forecasts financial performance after receiving information from other departments.

Human resources: The human resources, or personnel, department finds and hires people and administers **sick leave** and retirement matters. It is also concerned with compensation levels, professional development, employee relationship, and government regulations.

Information systems (IS): The IS department manages the organization's computer-based systems and plans for and purchases new ones.

Large organizations traditionally have three levels of management—strategic management, **tactical** management, and operational management. These levels can be shown on an organization chart, a **schematic** drawing showing the hierarchy of formal relationships among an organization's employees. Managers on each of the three levels have different levels of responsibility and are therefore required to make different kinds of decisions (See Figure 10-1 again).

Strategic-level management: Top managers are concerned with **long-range**, or strategic, planning and decisions. This top level is headed by the chief executive officer (CEO) along with several vice presidents or managers with such titles as chief financial officer (CFO), chief operating officer (COO), and chief information officer (CIO). Strategic decisions are complex decisions rarely based on predetermined routine procedures; they involve the **subjective** judgment of the decision maker. For instance, strategic decisions relate to how growth should be **financed** and what new markets should be **tackled** first. Determining the company's 5-year goals, evaluating future financial resources, and formulating a response to competitors' actions are also strategic decisions.

Tactical-level management: Tactical, or middle-level managers, make tactical decisions to implement the strategic goals of the organization. A tactical decision is made without a base of clearly defined informational procedures; it may require detailed analysis and computations. Examples of tactical-level managers are plant manager, division manager, sales manager, branch manager, and director of personnel.

Operational-level management: Operational, or low-level (supervisory level), managers make operational decisions—predictable decisions that can be made by following well-defined sets of routine procedures. These managers focus **principally** on supervising non-management employees, monitoring day-to-day events, and taking corrective action where necessary. An example of an operational-level manager is a warehouse manager in charge of inventory restocking.

Note that, because there are fewer people at the level of top management and many people at the bottom, this management structure resembles a **pyramid**. It's also a hierarchical structure because most of the power is concentrated at the top.

Words**sizable**

['saɪzəbl] adj. 相当大的, 很大的

famine

['fæmɪn] n. 饥荒, 奇缺

end

[end] n. 目的, 目标

promotion

[prəʊ'məʊʃən] n. (商品等的) 宣传, 推销 (活动)

price

[praɪs] v. 给……定价

package

['pækɪdʒ] v. 包装 (商品)

paycheck

['peɪtʃek] n. 薪金, 工资, (支付薪金或工资的) 支票

administer

[əd'mɪnɪstə] v. 掌管, 管理

tactical

['tæktɪkəl] adj. 战术的

schematic

[ski'mætɪk] adj. 示意性的, 图示的

long-range

['lɒŋ'reɪndʒ] adj. 长远的, 长期的

subjective

[sʌb'dʒektɪv] adj. 主观的, 个人的

finance

[faɪ'næns] v. 供给……经费, 负担经费

tackle

['tækl] v. 处理, 解决

principally

['prɪnsɪpli] adv. 主要地, 大部分地

pyramid

['pɪrəmid] n. 金字塔

Phrases

sick leave

(尤指拿病假工资的) 病假

Abbreviations

R&D

Research and Development | 研究与开发

Complex Sentences

Original: Information—whether computer-base or not—has to flow within an organization in a way that will help managers, and the organization, achieve their goals.

Translation: 信息——不论是否基于计算机——必须在组织内部以一种能够帮助管理者和组织达到其目标的方式流动。

Exercises

I. Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

___1. In the horizontal view, different departments of an organization are partitioned based on

different functions.

___ 2. There is a horizontal structure within each department of an organization.

___ 3. Strategic decisions are more long-term than tactical decisions.

___ 4. Tactical decisions don't require detailed analysis and computations.

___ 5. Well-defined sets of routine procedure can help to make predictable operational decisions.

II. Choose the best answer to each of the following questions.

1. Which statement is wrong about the organizational structure of a company according to this article?
 - (A) Information always flows from the top to the bottom within a pyramidal organization.
 - (B) The IS department is generally parallel to the marketing and sales department.
 - (C) Sales managers belong to the marketing department horizontally and tactical-level of management vertically.
 - (D) Non-management employees are supervised directly by managers on operational-level.
2. Which of the following is right about the responsibilities of different departments?
 - (A) The R&D department conducts basic research and makes investments.
 - (B) The production department relates the discoveries in research to the organization's current or new products.
 - (C) The marketing department spreads the services or goods to the public.
 - (D) The human resources department issues salaries.
3. Which of the following is wrong about the levels of management in an organization?
 - (A) Decisions made on lower levels are to implement the superior goals.
 - (B) Fewer people hold most of the power.
 - (C) Staff on each of levels has different levels of responsibility.
 - (D) Strategic decisions are made by the completely objective judgment of the superiors on the top level.

III. Translation

1. The human resources, or personnel, department finds and hires people and administers sick leave and retirement matters.
2. The IS department manages the organization's computer-based systems and plans for and purchases new ones.

➤ Section B: Google Glass 谷歌眼镜

Google Glass is an attempt to free data from desktop computers and portable devices like phones and tablets, and place it right in front of your eyes.

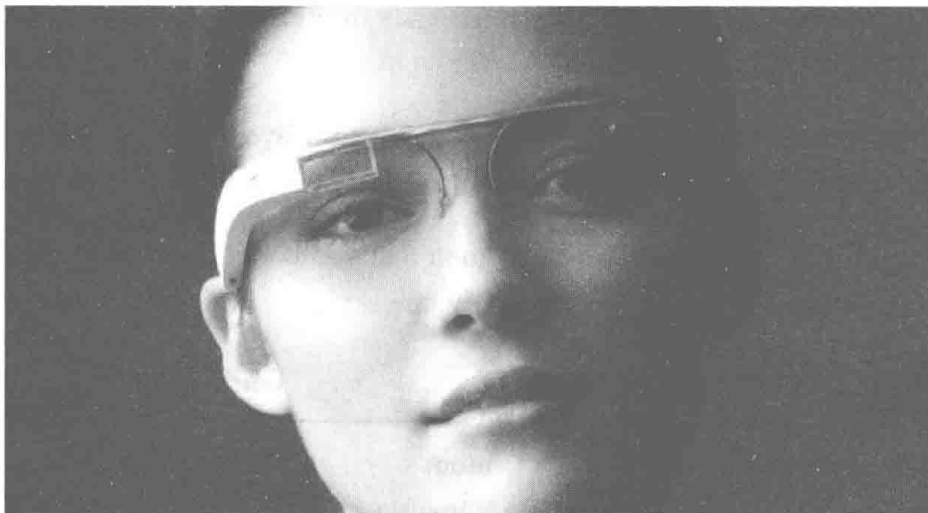
Essentially, Google Glass is a camera, display, touchpad, battery and microphone built into spectacle frames so that you can perch a display in your field of vision, film, take pictures,

search and translate on the go. [1]

The principle is one that has been around for years in science fiction, and more recently it's become a slightly **clunky** reality. In fact, the “heads-up display” putting data in your field of vision became a reality as early as 1900 when the **reflector** sight was invented.

Google Glass uses display technology instead to put data in front (or at least, to the upper right) of your vision **courtesy of** a **prism** screen. This is designed to be easily seen without **obstructing** your view. According to Google the display is “the equivalent of a 25-inch high definition screen from eight feet away”. There's no official word on **native resolution**, but 640×360 has been widely **mooted**.

Overlaying data into your vision has obvious benefits; many of which are already functional in Google Glass. Directions become more **intuitive** (although it sounds like there is no GPS on board so you will have to pair it with your phone), you can view real-time translations or transcriptions of what is being said, and you can scroll through and reply to messages—all **on the fly**.



The embedded camera obviously does not need a **viewfinder** because it is simply recording your **first-person** perspective, allowing you to take **snapshots** or **footage** of what you are actually seeing.

Any function that requires you to look at a screen could be put in front of you.

Controlling this data is the next **neat** trick. With a microphone and touchpad on one arm of the frame, you can select what you want to do with a brief gesture or by talking to the device, and Google Glass will interpret your commands.

Google Glass can also provide sound, with **bone-induction** technology confirmed. This vibrates your skull to create sound, which is both more **grisly** sounding and much less **cumbersome** than traditional headphones.

As well as Google's own list of features, the early apps for Google Glass provide a neat glimpse into the potential of the headset.

As well as photos and film—which require no explanation—you can use the Google hangout software to video conference with your friends and show them what you're looking at.

You'll also be able to use Google Maps to get directions, although with GPS absent from the spec list, you'll need to tether Glass to your phone.

To do that, Google offers the MyGlass app. This pairs your headset with an Android phone. As well as sharing GPS data, this means messages can be received, viewed on the display, and answered using the microphone and Google's voice-to-text functionality.

Google has given its Glass project a big boost by **snapping up** voice specialists DNNresearch.

That functionality will also bring the ability to translate the words being spoken to you into your own language on the display. Obviously you'll need a Wi-Fi connection or a **hefty** data plan if you're in another country, but it's certainly a neat trick if it works.

Third parties are also already developing some rather cool/scary apps for Google Glass—including one that allows you to identify your friends in a crowd, and another that allows you to dictate an E-mail.

The New York Times app gives an idea how news will be displayed when it's asked for: a headline, **byline**, appropriate image and number of hours since the article was published are displayed.

Other cool ideas include an air carrier's suggestion that you could have flight details **beamed** to you while you are waiting at the airport. Basically, the sky's the limit.

Words

spectacle

['spektəkl] n. 眼镜

frame

[freim] n. 眼镜框

perch

[p3:tʃ] v. 停留

clunky

['klʌŋki] adj. 粗笨的, 笨重的

reflector

[ri'flektə] n. 反光镜, 反射镜

prism

['prizəm] n. 棱镜

obstruct

[əb'strʌkt] v. 妨碍, 阻碍, 阻止

moot

[mu:t] v. 提出……供讨论, 提出(问题)

intuitive

[in'tuitiv] adj. 直觉的, 凭直觉获知的, 直观的

transcription

[træn'skripʃən] n. 录音, 翻译, 抄写

viewfinder

['viufaində] n. (照相机)取景器, 反光镜

snap

[snæp] n. 照片

footage

['fʊtɪdʒ] n. 连续镜头, 电影片段

neat

[ni:t] adj. 巧妙的, 简洁的, 简便易行的

Words**grisly**

['grizli] adj. 可怕的, 厉害的, 严重的

cumbersome

['kʌmbəsəm] adj. 复杂而低效的, 笨重的

hangout

['hæŋaʊt] n. 住所, 常去的地方

hefty

['hefti] adj. 相当多的, 异常大的

scary

['skeəri] adj. 引起恐慌的, 骇人的

byline

['baɪlaɪn] n. 署名

beam

[bi:m] v. 发送, 发射

Phrases

be courtesy of

由……提供

native resolution

标准分辨率

on the fly.

联机, 在线

first-person

第一人称

bone-induction

[医]骨诱导

snap up

抢购, 抢先弄到手

Complex Sentences

Original: Essentially, Google Glass is a camera, display, touchpad, battery and microphone built into spectacle frames so that you can perch a display in your field of vision, film, take pictures, search and translate on the go.

Translation: 本质上来说, 谷歌眼镜是一个加装了摄像机、显示器、触摸板、电池以及扩音器的眼镜框, 所以可以暂停视角、记录视频、拍照片、搜索及解说眼前的事物。

Exercises

I . Read the following statements carefully, and decide whether they are true (T) or false (F) according to the text.

- ___ 1. Google Glass is an attempt to free data from desktop computers and portable devices like phones and tablets, and place it right beside your ears.
- ___ 2. Wearing Google Glass can obstruct your view.
- ___ 3. Google Glass allows you to take snaps or footage of what you are actually seeing.
- ___ 4. One of the apps that are developing for Google Glass allows you to dictate an E-mail.
- ___ 5. Google Glass's display is "the equivalent of a 25-inch high definition screen from eighty feet away".

II. Choose the best answer to each of the following questions.

1. How many feet away does the Google Glass's display equal a 25-inch high definition screen from?
(A) 2
(B) 4
(C) 6
(D) 8
2. Which of software can be used to get directions for Google Glass?
(A) IBM DB2
(B) Google Maps
(C) Adobe Photoshop
(D) Microsoft Word
3. Essentially, which of the following is Google Glass built into spectacle frames except?
(A) Cup
(B) Microphone
(C) Display
(D) Camera

III. Translation

Directions become more intuitive (although it sounds like there is no GPS on board so you will have to pair it with your phone), you can view real-time translations or transcriptions of what is being said, and you can scroll through and reply to messages — all on the fly.

Part 3 Simulated Writing: Business E-mail

E-mail is an electronic, computer-assisted online communication tool. In the business world it is used to transmit virtually every type of correspondence the daily conduct of business requires. Simple messages, memos and letters, complex reports, tables of data, graphs and charts, blueprints, pictures, you name it. If it can be generated by, scanned into, or downloaded onto a computer, it can be electronically sent through cyberspace to another computer.

In this guide you will read about writing business E-mails and helpful tips on formatting business E-mail. Each section provides useful information to assist you in becoming more proficient at using E-mail to communicate in the business world.

Formatting Business E-mail

The format of business E-mail is very similar to that of a business memo. So similar, in fact, that the basic heading elements found in a business memo are programmed into the computer

generated template of every E-mail program. The communication role of business E-mail goes beyond that of a memo, however. In order not to overlook its versatility, the formatting elements of a business letter can be inserted manually into the body of an E-mail.

Business E-mail functions as both an internal and an external method of communication; its three main formatting elements are the heading, the body, and a signature block. Depending upon the nature of its correspondence and, at the discretion of the writer, business E-mail may also include a salutation and a complimentary close.

Business E-mail Heading

The heading of a business E-mail consists of up to six distinct information fields. They are located at the corner of the E-mail template, just below the tool bar.

The template itself appears automatically whenever you click on the New Mail, Reply, Reply to All, or the Forward button found on the tool bar of any E-mail program.

Each field in the template is designed to hold specific information, the definitions of which are preprinted on the left hand margin as follows:

- To:
- From:
- Cc:
- Bcc:
- Date:
- Subject:
- Attached:

Some of these fields are not always visible. The Bcc and Attached fields, for instance, are visible only when activated by the sender and, depending on the E-mail program, the From and Date fields may not be visible on the sender's template.

Business E-mail Body

The body of a business E-mail is no different than that of a business letter or memo. The one formatting distinction is that E-mail programs automatically format the body in single spaced, full block style. It's a design function of the program and meant to ensure that the text of an E-mail appears on the recipient's screen exactly as it does on the sender's.

Short paragraphs are the rule, particularly as online readers often just scan the text. Many monitors display twenty to twenty five lines at one time, making shorter paragraphs more suitable. Besides, long paragraphs are hard on the eyes and more difficult to read.

Always be considerate of your reader's time; an E-mail that goes on for more than two pages may be better off as a letter or memo. Delivered as an attachment, a lengthy letter or memo can

be printed and read when time permits.

Generally speaking, business E-mail is sent in plain text rather than HTML. Using different font faces, colors, sizes, and styles, such as bold or italic, is extremely useful when creating documents within a word processor, but they are problematic when sent between E-mail programs.

Besides the possibility of increasing the download time on your recipient's computer, HTML documents may not appear on one screen as they do on another. If you are unfamiliar with the distinction and are offered a choice, just select plain text and move on.

Business E-mail Signature Block

The signature block in a business E-mail does the work of the heading or letterhead found in the format of a business letter. In other words, it supplies the contact information belonging to the sender. This is the last item in an E-mail. It is always located on the left hand margin below the signature line.

A signature block should contain all the contact information a recipient might require in order to respond to an E-mail. It should begin with the Sender's Name, Title, and Business Organization. A Physical Location, Phone Numbers, E-mail Address, and Web site should follow.

Business E-mail Salutations and Complimentary Closes

Including a salutation and complimentary close in a business E-mail is governed by the same rules as those governing business letters and memos.

Formal expressions such as Dear Ms. Ortiz and Sincerely yours are suitable for letter style business E-mails addressed to individuals with whom you are unfamiliar. When business E-mail functions as a memo, on the other hand, a salutation and complimentary close should be omitted altogether. More often than not, the salutation and complimentary close will be written informally, particularly after the protocols of an initial contact have been observed.

A Sample Business E-mail

To: Expansion Community: itpartners@expansion.com, winpartners@expansion.com

From: Software Release Team

Cc: itag@expansion.com

Bcc: is&t@expansion.com

Date: Monday, February 3, 2014

Subject: SmartMessenger 11.0i Release Announcement

Attached: Brochure of SmartMessenger Enterprise 11.0i

Good Afternoon,

On behalf of the IS&T SmartMessenger Release Team, I am pleased to announce the release of SmartMessenger Enterprise 11.0i. SmartMessenger is an instant messaging program that runs on

the Windows 7/8 client and Windows 2008 and 2012 server platforms. SmartMessenger Enterprise 11.0i contains many enhancements and new features, including the ability to share files among group members and to subscribe E-mails from particular departments.

How to Obtain:

Expansion Company possesses a site license for SmartMessenger, so the program is available to Expansion Company staff and affiliates free of charge. You can download the product from the Expansion Company IS&T Windows Software Site:

<https://web.expansion.edu/software/win.html>.

Please note that you need a current personal certificate to download this software. If you do not have a current personal certificate, then you can obtain one from <https://www.expansion.com/>.

Getting Help:

The SmartMessenger at Expansion Company page is located at:

<http://itinfo.expansion.com/product.php?vid=644>

If you have a question or need assistance, please contact the Computing Help Desk at: computing-help@expansion.com.

All the Best,

Kevin Li

SmartMessenger 11.0i Product Release Coordinator

Software Release Team

Information Services and Technology Department

Expansion Company

Address: Zhong Guancun Road, Haidian District, Beijing, China

Code: 100000

Phone: 8610-62310000

Fax: 8610-62311001

E-mail: kevinli@expansion.com

Web site: <http://expansion.com>

Exercises

There are two scenarios below, please choose one and write an E-mail.

Scenario 1: An E-mail to your manager about the work progress in the past week while let all of your team members know.

Scenario 2: An E-mail to your manager about your business trip plan next week.

Glossary (词汇表)

A

accelerate	[æk'seləreit] v.	(使) 加快, (使) 增速
acceleration	[æk,selə'reiʃən] n.	加速, 加速升级
accelerometer	[æk,selə'rɒmitə] n.	加速计
acceptance testing		验收测试
accommodate	[ə'kɒmədeɪt] v.	容纳
account	[ə'kaunt] n.	客户
account for		(在数量、比例方面) 占
acronym	['ækrənɪm] n.	只取首字母的缩写词
ad hoc	['æd'hɒk] adv.	特别
adapt	[ə'dæpt] v.	使适合, 使适应
adapt oneself to		使适应
adaptive	[ə'dæptɪv] adj.	适应的
additive	['ædɪtɪv] adj.	附加的
address	[ə'dres] v.	处理, 满足
adjust	[ə'dʒʌst] v.	调节, 调整
administer	[əd'mɪnɪstə] v.	掌管, 管理
advance	[əd'vɑ:ns] v.	提升, 提高
advise	[əd'vaɪz] v.	忠告, 给……出主意, 劝告
advocate	['ædvəkeɪt] v.	提倡, 主张
aeronautics	[eərə'nɔ:tɪks] n.	航空学
afar	[ə'fɑ:] adv.	从远处
agent	['eɪdʒənt] n.	使然力 (指引起一定作用的人或其他因素)
agilist	['ædʒaɪlɪst] n.	敏捷主义者, 机敏者
agree to		同意
ain't	[eɪnt] prep.	(= are not, am not, is not) 不是
airfare	['eəfeə] n.	飞机票价
albeit	[ɔ:l'bi:ɪt] conj.	虽然
algorithm	['ælɡərɪðəm] n.	算法
alpha testing		α 测试

amendment	[ə'mendmənt] n.	修正案, 修正条款
among other things		其中
analogy	[ə'nælədʒi] n.	类比, 类似
anarcho-capitalist		无政府资本主义者
animation	[ˌæni'meɪʃən] n.	动画
animator	[ˈænimeɪtə] n.	漫画家, 动画片绘制者
anomaly	[ə'nɒməli] n.	异常, 反常
anticipate	[æn'tisipeɪt] v.	预期
appeal to		对……有吸引力
apply to		应用
appreciate	[ə'pri:ʃieɪt] v.	赏识, 欣赏, 重视
approach	[ə'prəʊtʃ] v.	提议
approve	[ə'pru:v] v.	批准, 通过
architect	[ˈɑ:kitekt] n.	架构师, 设计师
architectural design		体系结构设计, 概要设计
archive	[ˈɑ:kai v] v.	存档
argue	[ˈɑ:gju:] v.	主张, 认为
arise	[ə'raɪz] v.	出现, 产生
artist	[ˈɑ:tɪst] n.	(某方面的) 能手, 大师, 画家, 艺术家
as	[æs] pron.	同样
assembly	[ə'sembli] n.	汇编
assimilate	[ə'simileɪt] v.	吸收
audit	[ˈɔ:dit] n.	审核, 查证
avalanche	[ˈævələ:ntʃ] n.	纷至沓来, 蜂拥而至, 雪崩
axiom	[ˈæksiəm] n.	公理, 格言, 规则

B

backup	[ˈbækʌp] adj.	后备的
badge	[bædʒ] n.	徽章, 证章, 标记
base station		基站
be concerned with		参与, 关于
be courtesy of		由……提供
be derived from		源于, 得自
be grateful for		为……而感谢
be involved in		参与
be made of		用……制成
be suited for		适合于

be tied to		被束缚
beam	[bi:m] v.	发送, 发射
beg	[beg] v.	请求, 恳求
beta testing		β 测试
Bitcoin		点对点 (peer-to-peer) 基于网络的匿名数字货币
black-box testing		黑盒测试
blueprint	['blu:print] n.	蓝图, 计划
board	[bɔ:d] n.	理事会, 董事会
bogeyman	['bəʊgimæn] n.	可怕 (或可憎) 的人或物, 妖怪, 精灵
bone-induction		[医]骨诱导
book	[buk] v.	登记, 预订
bottom-up approach		自底向上方法
branch	[brɑ:ntʃ] v.	分支
breakup	['breikʌp] n.	分解
browser	[brauzə] n.	浏览器
bring to bear		应用, 使用
bug	[bʌg] n.	程序缺陷
build	[bild] n.	编好的程序
built-in		嵌入的, 固定的
business	['biznis] n.	企业, 商业, 交易, 生意, 事务, 业务, 职业, 行业
buzz	[bʌz] n.	时髦的 (词语、想法或活动)
by the way		顺便
byline	['bailain] n.	署名

C

cache	[kæʃ] v.	把 (数据) 存入高速缓冲存储器, 高速缓存
call center		呼叫中心
capture	['kæptʃə] v.	获得, 捕获
cardinal	['kɑ:dinəl] adj.	主要的, 最重要的
catastrophically	[kætə'strɒfikəli] adv.	灾难性地
categorical	[kæti'gɔrikəl] adj.	分类的, 按类别的
cause-effect		因果
cell towers		基站, 基地台
cellular	['seljʊlə] adj.	蜂窝状的

chaos	['keɪɒs] n.	混乱
check in		登记, 报到
cheer	[tʃiə] n.	欢呼, 喝彩
chief	[tʃi:f] n.	长官, 首领
chip	[tʃɪp] n.	芯片
chronic	['krɒnɪk] adj.	慢性的, 长期的
churn	[tʃɜ:n] n.	粗制滥造, 搅拌
cite	[saɪt] v.	引用
class	[klɑ:s] n.	类
classify as		把……归类为
cloud computing		云计算
clunky	['klʌŋki] adj.	粗笨的, 笨重的
cluster	['klʌstə] n.	群, 簇
code	[kəʊd] v.	编码
cognitive	['kɒgnɪtɪv] adj.	认知的, 认识的
cohesion	[kəʊ'hi:ʒən] n.	内聚
cohesiveness	[kəʊ'hi:sɪvnɪs] n.	黏结性, 内聚性
column	['kɒləm] n.	专栏 (文章)
combine with		使……与……联合
come out of		出自
come to a conclusion		得出结论, 告一段落
commence	[kə'mens] n.	开始
comment	['kɒment] n.	注释, 注解
commission	[kə'mɪʃən] n.	佣金, 回扣, 代办费, 手续费
common sense		常识 (尤指判断力)
communicator	[kə'mju:nɪkeɪtə] n.	传播者, 通信员
compact	[kəm'pækt] adj.	简明的, 紧凑的
compare to		把……比作
competent	['kɒmpɪtənt] adj.	有能力的, 胜任的
compiler	[kəm'paɪlə] n.	编译器
compromise	['kɒmprəmaɪz] n.	妥协, 折中
concern	[kən'sə:n] v.	担心, 忧虑
concurrency	[kən'kʌrənsɪ] n.	并行
conduct	[kən'dʌkt] v.	实施, 进行
configuration	[kən'fɪgju'reɪʃən] n.	配置
configuration items		配置项
conform to		遵守
confront	[kən'frʌnt] v.	面对, 碰到
Congress	['kɒŋɡres] n.	(美国) 国会, 议会

consciously	['kɒnʃəsli] adv.	有意识地
consensus	[kən'sensəs] n.	(意见等) 一致
consistency	[kən'sistənsi] n.	一致性, 连贯性
construct	[kən'strʌkt] n.	结构体
contemplate	['kɒntempleit] v.	思考, 预期
contemporary	[kən'tempərəri] adj.	属于同时期的, 同时代的
context	['kɒntekst] adj.	上下文, 环境, 背景
contract	[kən'trækt] v.	承包, 订契约
converge	[kən'və:dʒ] v.	会聚
convergence	[kən'vɜ:dʒəns] n.	一体化, 集中, 收敛
coordinate	[kəu'ɔ:dɪnɪt] v.	调整, 整理
cope with		对付, 应付
correspondence	[kɔrɪs'pɒndəns] n.	通信
cost-effective	[kɒsti'fektiv] adj.	有成本效益的, 划算的
couple	['kʌplɪŋ] n.	耦合
course	[kɔ:s] n.	过程, 进程
craft	[krɑ:ft] v.	精心策划, 制作
criterion	[krai'tiəriən] n.	标准, 准则 (复数为 criteria)
cross-reference	['krɒsrɪfərəns] v.	前后对照
crunch	[krʌntʃ] v.	处理, [俚]运行
cryptography	[krip'tagrəfi] n.	密码使用法, 密码系统, 密码术
culminate	['kʌlmineɪt] v.	告终, 达到顶点
cumbersome	['kʌmbəsəm] adj.	复杂而低效的, 笨重的

D

data warehouse		数据仓库
daunt	[dɔ:nt] v.	沮丧, 使气馁
deal with		处理
debug	[di:'bʌg] v.	调试
deduce	[di'dju:s] v.	推论, 演绎出
deem	[di:m] v.	认为
defect	[di'fekt] n.	缺陷
deflation	[di'fleɪʃən] n.	通货紧缩, 降阶
degrade	[di'greɪd] v.	降级, 退化
delivery	[di'livəri] n.	交付
demonstrate	['demənstreɪt] v.	证明, 论证
depict	[di'pɪkt] v.	描述, 描写

deployment	[di'plɔimənt] n.	部署
deposit	[di'pɒzɪt] n.	押金, 预付金
deserve	[di'zə:v] v.	值得, 应得
design constraint		设计约束
desire	[di'zaɪə] v.	要求, 需要
desk-checking		部件检验
detailed design		详细设计
discipline	['disiplin] n.	学科
disruptive	[dis'rʌptɪv] adj.	颠覆性的, 破坏的
distinct	[dis'tɪŋkt] adj.	不同的, 有区别的
distinctive	[dis'tɪŋktɪv] adj.	区别性的, 特殊的
dive into		投入, 钻研
diversity	[dai'və:sɪti] n.	差异, 多样性
divide-and-conquer		分而治之
document	['dɒkjumənt] v.	纪实性地描述
download	['daʊnləʊd] v.	下载
downside	['daʊnsaɪd] n.	缺点, 负面
due to		由于

E

egoless	['i:gəʊlɪs] adj.	不自负的, 非自我的
elaboration	[ɪ,læbə'reɪʃən] n.	精化
elevate	['elɪveɪt] v.	提高, 提升
elicitation	[ɪ,lɪsi'teɪʃən] n.	导出, 启发
elusive	[ɪ'lju:sɪv] adj.	难于被发现的
embed	[ɪm'bed] v.	嵌入
embrace	[ɪm'breɪs] v.	掌握, 接受, 拥抱
encapsulation	[ɪn,kæpsju'leɪʃən] n.	封装
encompass	[ɪn'kʌmpəs] v.	包含, 包括
end	[end] n.	目的, 目标
end up		(以……) 结束, 最终成为[变得], 最后处于
endeavor	[ɪn'devə] n.	努力, 尽力
end-stage		末期
energetic	[ɪnd'ɜ:ɡetɪk] adj.	精力充沛的
epsilon	[ep'saɪlən] n.	小的正数
essence	['esns] n.	本质, 实质

exabyte	[ˈeksə,bait] n.
except for	
exercise	[ˈeksəsaɪz] v.
exert	[ɪgˈzə:t] v.
exhaustively	[ɪgˈzɔ:stɪvli] adv.
exploitable	[iksˈplɔɪtəbl] adj.
Extreme Programming	

艾字节
除……之外
执行
尽（力），施加（压力等）
详尽地，彻底地
有利可图的，可开发的，可利用的
极限编程

F

fab	[fæb] n.
fabrication	[ˌfæbrɪˈkeɪʃən] n.
facilitate	[fəˈsɪlɪteɪt] v.
fall off	
famine	[ˈfæmɪn] n.
fan-in	[ˈfænɪn] n.
fan-out	[ˈfæn,aʊt] n.
feature phone	
finance	[faɪˈnæns] v.
finger-pointing	
first-person	
fit currency	
fit for	
fix	[fiks] v.
flexibility	[ˌfleksəˈbɪlɪti] n.
flowchart	[ˈfləʊtʃɑ:t] n.
fluid	[ˈflu(:)ɪd] adj.
flying	[ˈflaɪɪŋ] adj.
footage	[ˈfʊtɪdʒ] n.
for good	
form factor	
frame	[freɪm] n.
freakish	[ˈfri:kɪʃ] adj.
free of	
fresh	[freʃ] adj.
from end to end	
fruition	[fruˈɪʃən] n.
fruitless	[ˈfru:tɪlis] adj.

代工厂
制造，生产
使容易，使便利，促进
脱落，掉队
饥荒，奇缺
扇入，输入端
扇出，输出端
功能手机
供给……经费，负担经费
指责，责难
第一人称
可使用的通货
适合
使固定，确定
弹性，适应性，灵活性
流程图
变化的，流动的，液体的
会飞的，飞行的
连续镜头，电影片段
永远
（电子产品等的）物理尺寸和形状，规格
眼镜框
反常的，奇怪的
没有……的
新到的，无经验的
从头到尾地
实现，结果实
徒劳的，无用的

function	[ˈfʌŋkʃən] n.	函数
function-oriented		面向功能的
funding	[ˈfʌndɪŋ] n.	提供资金
furthermore	[fɜːðə'mɔː] adv.	此外, 而且

G

Gantt Chart		甘特图
geek	[gi:k] n.	对计算机痴迷的人
generation	[dʒenə'reiʃən] n.	一代, 阶段
genomic	[dʒi:'nəʊmɪk] adj.	染色体组的
get one's hands on		把……弄到手
get used to		习惯
give up		放弃
given	[ˈɡɪvən] n.	假设
go under		归为 (某一类事物) 中
go up		上升, 增长
golf	[ɡɒlf] v.	打高尔夫球
grey-box testing		灰盒测试
grid computing		网格计算
grisly	[ˈɡrizli] adj.	可怕的, 厉害的, 严重的

H

halve	[hɑ:v] v.	把……分成两半, 把……减半
hand off		手递手传
hands-on	[ˈhændz'ɒn] adj.	亲身实践的, 实际动手做的
hangout	[ˈhæŋaʊt] n.	住所, 常去的地方
head	[hed] v.	负责, 领导
headquarter	[ˌhed'kwɔ:tə] v.	总部设于……总公司设于……
hefty	[ˈhefti] adj.	相当多的, 异常大的
help desk		(向网络用户提供信息和技术支持的) 用户服务部
heuristic	[hjuə'ristɪk] adj. & n.	启发式的, 启发式的论据
hierarchical	[ˌhaɪə'rɑ:kɪkəl] adj.	分等级的, 层次的
high season		旺季
high-end	[haɪ end] adj.	(尤指电子产品) 最高档的, 高端的, 最昂贵的

highlight	['haɪlaɪt] v.	强调, 使突出
hitherto	[hɪðə'tu:] adv.	迄今, 至今
homogeneous	[hə'məʊ'dʒi:njəs] adj.	相同特征的, 同性质的
hook up with		沟通, 交流
identify with		视……为一体, 认同
impact	['ɪmpækt] n.	影响, 效果
impersonal	[ɪm'pə:sənəl] n.	非个人的
implicitly	[ɪm'plɪsɪtli] adv.	不言明地, 默认地
impose	[ɪm'pəʊz] v.	强制实行
in addition		此外
in charge of		负责, 领导
in concert		异口同声地, 齐声
in detail		详细地
in exchange for		作为……的交换, 用……交换
in moderation		适中地
in question		考虑之中的, 被谈论着的
in that		因为
incentive	[ɪn'sentɪv] n.	刺激, 诱因, 鼓励, 动机
inception	[ɪn'sepʃən] n.	开端
incorporate	[ɪn'kɔ:pəreɪt] v.	加入, 包含
indispensable	[ɪndɪ'spensəbl] adj.	不可缺少的, 必需的
inform	[ɪn'fɔ:m] v.	通知, 告知
infrastructure	['ɪnfreɪstrʌktʃə] n.	基础设施, 基础建设
ingredient	[ɪn'ɡri:diənt] n.	成分, 因素
inherent	[ɪn'hɪərənt] adj.	固有的, 内在的, 与生俱来的
inheritance	[ɪn'herɪtəns] n.	继承
in-house	[ɪn'haʊs] adj.	内部的
injection-mold		注塑模具
institution	[ɪnstɪ'tju:ʃən] n.	习俗, 制度
intangible	[ɪn'tændʒəbl] adj.	无形的
integrated circuits		集成电路
integrator	['ɪntɪɡreɪtə] n.	综合者, 合成者
intensive	[ɪn'tensɪv] adj.	密集的, 加强的, 集中的
interested	['ɪntrɪstɪd] adj.	有利害关系的, 有份儿的, 有关的
internship	['ɪntə:nʃɪp] n.	实习生职位, 实习期

interoperability	[ˈɪntə,ɒpərəˈbɪləti] n.	互用性, 协同工作的能力
interpersonal	[ˌɪntəˈpɜːsənl] n.	人际(关系)的
interpreter	[ɪnˈtɜːprɪtə] n.	解释程序, 解释器
intimate	[ˈɪntɪmɪt] adj.	精通的, 基本的
introduce	[ˌɪntrəˈdʒuːs] v.	引入
intuitive	[ɪnˈtʃuɪtɪv] adj.	直觉的, 凭直觉获知的, 直观的
invoke	[ɪnˈvəʊk] v.	调用
isolation	[ˌaɪsəʊˈleɪʃən] n.	隔离, 孤立
issue	[ˈɪʃɪʊ] n.	问题
iteration	[ˌɪtəˈreɪʃən] n.	迭代, 反复
iterative	[ˈɪtəətɪv] n. & adj.	迭代(的)
iterative and incremental approach		迭代和增量方法

J

Jackson's structure diagram		Jackson 结构图
jaw-breaking		难发音的
jell	[dʒel] n.	果冻
judge	[dʒʌdʒ] v.	判断, 评价
juggernaut	[ˈdʒʌgənɔːt] n.	骇人的(毁灭)力量
just about		几乎, 几近, 大约, 马上

K

keep in mind		记住
knit	[nɪt] v.	接合

L

law	[lɔː] n.	规则, 法则
lay out		划定(路线), 布置, 安排
lead to		导致
leading-edge		领先优势的
ledger	[ˈledʒə] n.	总账, 账簿
letter	[ˈletə] n.	字母
liaison	[liˈeɪzɔːn] n.	联络员

libertarian	[,libə'teəriən] n.	(思想或行动等的) 自由论者
literally	['litərəli] adv.	真实地, 确切地
literature	['litəritʃ ə] n.	著作, 文献
lithography	[li'θɔgrəfi] n.	平版印刷术
live	[laiv] adj.	生动的
log	[lɒg] v.	记录
long-range	['lɒŋ'reindʒ] adj.	长远的, 长期的
loop	[lu:p] n.	循环
low season		淡季
lunar	['lu:nə] adj.	月球的

M

maintainer	[men'teinə] n.	维护人员
major	['meɪdʒə] adj.	较大的, 较重要的
make sense		理解, 有意义, 是明智的
mammoth	['mæməθ] adj.	庞大的, 巨大的
management	['mænɪdʒmənt] n.	管理人员, 管理部门
manifest	['mænɪfest] v.	显示, 表明
manifesto	[mæni'festəu] n.	宣言, 声明
margin	['mɑ:dʒɪn] n.	利润
markedly	['mɑ:ktli] adv.	显著地, 明显地
Markup language		标识语言
master	['mɑ:stə] adj.	支配的, 总的, 最重要的
mayorship	['meəʃɪp] n.	市长的职位
mechanism	['mekənɪzəm] n.	机制
meet	[mi:t] v.	满足 (需求、要求等)
memory reference		存储器参量, 存储器参考, 内存引用
mental	['mentl] adj.	智力的, 脑力的
meta-information		元信息
metalanguage	['metə,læŋɡwɪdʒ] n.	元语言
metaphorical	[.metə'fɔrɪkəl] adj.	隐喻性的, 比喻性的
milestone	['maɪlstəʊn] n.	里程碑
mine	[maɪn] v.	开采
minicomputer	['mɪnɪkəm,pju:tə] n.	小型计算机
minimalist	['mɪnɪməlist] n.	最低限要求者
minor	['maɪnə] adj.	较小的, 次要的
mint	[mɪnt] adj.	崭新的, 完美的, 未损坏的

mitigation	[ˌmitiˈgeɪʃən] n.	缓解, 减轻
model	[ˈmɒdl] v.	建立模型
modeling-oriented		面向建模的
moderation	[ˌmɒdəˈreɪʃən] n.	适度, 节制
modularity	[ˌmɒdjuˈlærɪti] n.	模块性
momentum	[məʊˈmentəm] n.	动力, 冲力, 势头
moot	[mu:t] v.	提出……供讨论, 提出 (问题)
morale	[məˈrɑ:l] n.	士气, 民心
moreover	[məʊˈrəʊvə] adv.	而且, 此外
motivate	[ˈməʊtɪveɪt] v.	激发……的积极性
mutant	[ˈmju:tənt] n.	变体
mutation	[mju:ˈteɪʃən] n.	突变, 改变
mythical	[ˈmiθɪkəl] adj.	神话的, 虚构的

N

narrative	[ˈnærətɪv] adj.	叙述的
native applications		本地应用
native resolution		标准分辨率
neat	[ni:t] adj.	巧妙的, 简洁的, 简便易行的
negotiation	[niˌgəʊʃiˈeɪʃən] n.	协商
neighborhood		邻域细分
subdivision		
newscaster	[ˈnju:z,kɑ:stə] n.	(电台或电视的) 新闻播音员
no longer		不再
nonsensical	[nɒnˈsensɪkl] adj.	无意义的, 荒谬的, 愚蠢的
not yet		还没有
notable	[ˈnəʊtəbl] adj.	著名的
notably	[ˈnəʊtbli] adv.	尤其, 特别地
notation	[nəʊˈteɪʃən] n.	符号
note	[nəʊt] v.	指出, 对……加注释, 注意, 记录
notion	[ˈnəʊʃən] n.	概念, 观念, 看法

O

object	[ˈɒbdʒɪkt] n.	对象
object-oriented		面向对象的
obstacle	[ˈɒbstəkl] n.	障碍, 干扰

obstruct	[əb'strʌkt] v.	妨碍, 阻碍, 阻止
obviate	['ɒbvieɪt] v.	排除, 避免
omission	[əu'mɪʃən] n.	遗漏
on demand		一经要求, 点播
on the fly		联机, 在线
on the go		正在进行中, 忙碌中
one-off	['wʌnɒf] adj.	一次性的
online	['ɔ:n.laɪn] adj.	联机的, 在线的
operate on		产生作用
order	['ɔ:də] n.	订购, 订单
order of magnitude		数量级
outsource	['aʊtsɔ:s] v.	外包 (工程)
outsourcing	['aʊt.sɔ:siŋ] n.	外购, 外包
overriding	[əʊvə'raɪdɪŋ] adj.	首要的, 压倒一切的
oversee	['əʊvə'si:] v.	管理, 监督
oversight	['əʊvəsait] n.	疏忽, 漏失

P

package	['pækɪdʒ] v.	包装 (商品)
paper	['peɪpə] n.	论文, 文章
partition	[pɑ:'tɪʃən] v.	划分, 把……分成部分
pass down		把……传下去
patch	[pætʃ] n.	补丁 (程序)
pattern language		模式语言
pay a visit to		访问, 参观
paycheck	['peɪtʃek] n.	薪金, 工资, (支付薪金或工资的) 支票
peculiarly	[pi'kju:liəli] adv.	特有地, 特别地
peer-to-peer	['piətəpiə] adj.	对等
perch	[pɜ:tʃ] v.	停留
perfect	['pə:fɪkt] v.	使完善, 改进
periodic	[piəri'ɒdɪk] adj.	周期的, 定期的
pertain	[pə'tein] v.	关于, 有关
petabyte	['petəbait] n.	拍 (10^{15}) 字节
pick up		(尤指偶然地、无意地、容易地) 获得
pinpoint	['pɪnpɔɪnt] v.	精确地找到, 准确描述
pipeline	['paɪpleɪn] v.	传递
pixel	['pɪksl] n.	(显示器或电视机图像的) 像素

platform fragmentation		平台的碎片
plug	[plʌg] n.	插头, 插销
plus	[plʌs] conj.	并且, 而且
poke fun at		取笑, 调笑, 作弄, 戏弄
polymorphism	[ˌpɒli'mɔːfɪzəm] n.	多态
pool	['puːl] n.	资源的集合, 联营
populate	['pɒpjuleɪt] v.	构成, 填充
pose	[pəʊz] v.	引起, 造成
post	[pəʊst] v.	公布, 发布
power	['paʊə] v.	促进, 推动
practice	['præktɪs] n.	实践, 实行
practitioner	[præk'tɪʃənə] n.	从业者, 实践者
preassemble	['pri:ə'sembl] v.	预装(配), 预先安装
predator	['predətə] n.	以掠夺为生的人, 食肉动物
predatory	['predətəri] adj.	掠夺性的
predict	[pri'dɪkt] v.	预知, 预言
presence	['prezns] n.	势力
prevalent	['prevələnt] adj.	流行的, 盛行的, 普遍存在的, 普遍发生的
price	[praɪs] v.	给……定价
principally	['prɪnsɪpli] adv.	主要地, 大部分地
prioritize	[praɪ'ɔːtaɪz] v.	把……区分优先次序
prism	['prɪzəm] n.	棱镜
private cloud		私有云
productivity	[ˌprɒdʌk'tɪvɪti] n.	办公软件, 生产率, 生产力
profile	['prəʊfaɪl] v.	用数据图表(或曲线图)表示
profile	['prəʊfaɪl] n.	人物简介, 轮廓, 侧面
profiler	['prəʊfaɪlə] n.	曲线图, 概览图
profitability	[ˌprɒfɪtə'bɪlɪti] n.	收益性, 利益率
prolong	[prə'lɒŋ] v.	延长
prominent	['prɒmɪnənt] adj.	著名的
promotion	[prəʊ'məʊʃən] n.	(商品等的)宣传, 推销(活动)
proponent	[prəʊ'pəʊnənt] n.	支持者, 倡导者
proportion	[prəʊ'pɔːʃən] n.	部分, 比例
prospector	['prɒspektə] n.	勘探者, 探矿者
prototype	['prəʊtəʊtaɪp] n.	原型
provocative	[prəʊ'vɒkətɪv] adj.	引起争论(议论, 兴趣等)的
pseudocode	['psjuːdəʊkəʊd] n.	伪代码
public cloud		公有云

pursue	[pə'sju:] v.	从事, 进行
put into use		应用, 使用
put through		完成
put together		把……放在一起
pyramid	['pirəmid] n.	金字塔

Q

quasi-legal	准法律的, 准合法的
-------------	------------

R

rancher	['ræntʃə] n.	大牧场主, 大农场工人
rank	[ræŋk] n.	等级
reading through		通读
recipe	['resipi] n.	诀窍
recognize	['rekəgnaiz] v.	认可, 承认
reflector	[ri'flektə] n.	反光镜, 反射镜
refrains from		制止, 控制, 克制不要
refund	[ri:'fʌnd] v.	退还, 偿还
regime	[rei'ʒi:m] n.	体制, 体系
regression testing		回归测试
relate to		与……有关
release	[ri'li:s] v.	发布, 公开发行
reliability	[ri,laɪə'biliti] n.	可靠性
relinquish	[ri'liŋkwɪʃ] v.	放弃, 放手, 让渡
reminder	[ri'maɪndə] n.	提醒, 提示
reminiscent of		表明……的, 暗示……的
render	['rendə] v.	着色, 渲染
replicate	['replikɪt] v.	复制
repository	[ri'pɒzə'tɔ:ri] n.	仓库
resolution	[rezə'lʊʃən] n.	分辨率, 决心, 解决, 坚决
result from		由……引起
result in		导致
reuse	['ri:'ju:z] n.	复用
reveal	[ri'vi:l] v.	展现, 显示, 暴露
reward	[ri'wɔ:d] n.	报答, 报酬
rich	[ritʃ] adj.	宝贵的

rightly	['raitli] adv.	确实地
rigorous	['rigərəs] adj.	精确的, 严密的
ring out		(根据考勤钟) 打卡下班
ripple	['ripl] v.	起伏波动
road map		线路图, (一步一步的) 详尽计划
robust	[rəu'bʌst] adj.	强壮的, 精力充沛的
rollout	['rəul,aʊt] n.	首次展示, (产品或服务的) 正式推出
rotate	[rəu'teɪt] v.	轮流, 交替
rough	[rʌf] adj.	初步的, 粗略的
round down		四舍五入
routine	[ru:'ti:n] n.	程序
rule of thumb		单凭经验的方法

S

sanction	['sæŋkʃən] n. & v.	批准, 认可, 同意
savvy	['sævi] adj.	有经验的, 有见识的, 具有实际知识的
scalability	[skeilə'biliti] n.	可扩展性, 可伸缩性, 可量测性
scam	[skæm] n.	骗局, 诡计
scammer	['skæmə] n.	[美俚] 骗子, 诈骗犯
scary	['skeəri] adj.	引起恐慌的, 骇人的
scenario	[si'na:riəu] n.	场景, 某一特定情节
schematic	[ski'mætik] adj.	示意性的, 图示的
screwdriver	['skru:draivə] n.	螺丝刀
script	[skript] n.	脚本
semicolon	['semi'kəulən] n.	分号
sentiment	['sentimənt] n.	意见, 观点, 情绪
serve as		充当, 用作
shift	[ʃift] v.	摆脱掉, 改变, 去掉, 换挡
short run		短期 (指企业无法改变生产规模的较短期限)
show up		显而易见, (使) 显现出来
sick leave		(尤指拿病假工资的) 病假
sift	[sift] v.	详审, 精选
significant	[sig'nifikənt] adj.	意义重大的, 重要的
sizable	['saizəbl] adj.	相当大的, 很大的
skyrocket	['skai'rɒkit] v.	突升, 猛涨
smoke test		冒烟测试

smoothly	['smu:ðli] adv.	平稳地
snap	[snæp] n.	照片
snap up		抢购, 抢先弄到手
software crisis		软件危机
sophomore	['sɒfəmɔ:] n.	大学二年级学生
spawn	[spɔ:n] v.	造成, 大量产生
spearhead	['spiəhed] v.	当……的先锋, 带头
special	['speʃ(ə)l] n.	特价商品
specification	[ˌspesifi'keɪʃən] n.	说明书, 规范
spectacle	['spektəkl] n.	眼镜
speculator	['spekjuleitə] n.	投机商人
speedup	['spi:dʌp] n.	加速
spirited	['spiritid] adj.	热烈的
spot	[spɒt] v.	发现, 认出
staff	['stɑ:f] v.	配置职员
stage	[steɪdʒ] v.	展现, 举行
stakeholder	['steɪkhəʊldə] n.	利益相关者
Star Trek Replicator		星际迷航的复制
start-up	[stɑ:tʌp] n.	开始, 启动
state	[steɪt] v.	陈述, 说明
state-based		基于状态的
statement	['steɪtmənt] n.	语句
stem	[stem] v.	起源于
stem from		基于, 处于
stepwise	[stepwaɪz] adj.	逐步的
stress	[stres] n.	压力, 紧迫
stress testing		压力测试
subclass	['sʌbklɑ:s] n.	子类
subdivision	[ˌsʌbdɪ'vɪʒn] n.	细分, 一部, 分支
subjective	[sʌb'dʒektɪv] adj.	主观的, 个人的
sucker	['sʌkə] n.	对……入迷的人, 难以抗拒……的人
suffer	['sʌfə] v.	受损失, 受害
suffice	[sə'faɪs] v.	足够, 满足
suit	[sju:t] v.	适合, 合乎……的要求
suitably	['sju:təbli] adv.	相称地、适当地、适宜地
supervise	['sju:pəvaɪz] v.	监督, 管理
supplement	['sʌplɪmənt] v.	补充
surface	['sə:fɪs] v.	出现, 显露
symptomatic	[sɪmptə'mætɪk] adj.	有症状的

synchronization	[ˌsɪŋkrənai'zeɪʃən] n.	同步
synchronously	[ˌsɪŋkrənəsli] adv.	同时地, 同步地

T

tackle	['tækəl] v.	处理, 解决
tactical	['tæktikəl] adj.	战术的
tailor	['teɪlə] v.	剪裁, 适应
take action		采取行动, 着手
take advantage of		利用
take charge of		担任, 监管
take the lead		为首, 带头
take up		占 (空间、时间)
takes a view of		观察, 检查
talk over		商议, 讨论
technophile	['teknəʊ,faɪl] n.	技术爱好者
terabyte	['terəbaɪt] n.	太字节
test case		测试用例
tether	['teðə] v	束缚, 用绳拴住
that is to say		即, 换句话说
thin	[θɪn] adj.	稀少的
think about		考虑, 思考
thinker	['θɪŋkə] n.	思想者
threads	[θredz] n.	各组成部分
Thrive Movement		发展运动
throughout	[θru:'aut] prep.	遍及, 贯穿
throughput	['θru'put] n.	吞吐量, 生产量
top-down approach		自顶向下方法
trade-off	[treɪdɒf] n.	平衡, 折中
traditionalist	[trə'dɪʃənəlist] n.	传统主义者, 墨守成规者
trail	[treɪl] v.	(在比赛或其他竞赛中) 落后, 跟在……后面
trait	[treɪt] n.	特性
transaction	[træn'zækʃən] n.	事务
transcription	[træn'skripʃən] n.	录音, 翻译, 抄写
triangulate	[traɪ'æŋgjuleɪt] v.	对……作三角测量
trivial	['trɪviəl] adj.	琐细的, 价值不高的, 微不足道的
troubleshoot	['trʌblʃu:t] v.	故障检修, 问题解答

tumble	['tʌmbəl] v.	使倒下, 搅乱
turn-by-turn		路线规划, 建议路线, 分路段显示
tweet	[twi:t] n.	微博, 消息
typographical	[ˌtaɪpə'græfɪkəl] adj.	印刷上的, 排字上的

U

ultimately	['ʌltɪmətli] adv.	最后, 终于, 根本
uncover	[ʌn'kʌvə] v.	揭开, 揭露
under the supervision of		在……的监督下, 在……的指导下
underlie	[ʌndə'laɪ] v.	位于……之下, 构成……的基础
undertake	[ʌndə'teɪk] v.	着手做, 开始进行
unearth	[ʌn'ə:θ] v.	发现, 揭露
up to		(时间上) 一直到
upgrade	['ʌpgreɪd] n.	(尤指计算机设备等的) 升级
use case		用例
utility	[ju:'tɪlɪti] n.	公用事业, 公共事业设备

V

validation	[væli'deɪʃən] n.	确认, 正确性检测
value	['vælju:] v.	评价
verifiable	['verɪfaɪəbl] adj.	可证实的, 可检验的
verification	[ˌverɪfɪ'keɪʃən] n.	确认, 核查
verify	['verɪfaɪ] v.	检验, 核实
view	[vjʊ:] n.	视图, 观点
viewfinder	['vjʊfaɪndə] n.	(照相机) 取景器, 反光镜
virtually	['vɜ:tʃuəli] adv.	事实上, 实质上
vital	['vaɪtl] adj.	生死攸关的, 至关重要的
vulnerability	[ˌvʌlnərə'bɪləti] n.	弱点

W

wave	[weɪv] n.	汹涌的行动(或思想)态势, 涌现的人(或事物), 波浪
wear out		用坏
Webmaster	['web'mɑ:stə] n.	网站维护者, 网站管理员

well-defined	['weldi'faɪnd] adj.	明确的
well-suited	['wel'sutɪd] adj.	适当的, 便利的
whereabouts	['werəbaʊt] n.	下落, 行踪, 所在之处
white-box testing		白盒测试
Wikipedia		维基百科 (全书)
wing	[wɪŋ] n.	部门, 成员组织, 附属机构
with a view to		为了, 目的在于
with an eye to		着眼于
with respect to		就……而论, 关于
with the advent of		随着……到来
with the help of		借助
work out		解决, 算出
workload	['wɜ:kləʊd] n.	工作量

Y

Yoda	尤达 (《星球大战》里重要的人物)
------	-------------------

Abbreviation (缩略语表)

A

ACM	Association for Computing Machinery	美国计算机协会
ActiveX		微软公司倡导的网络化多媒体对象技术
ARPANET	Advanced Research Projects Agency Network	美国国防部高级研究计划管理局网络
ATM	Automatic Teller Machine	自动出纳机

B

BPO	Business Process Outsourcing	业务过程外包
-----	------------------------------	--------

C

CBD	Component-Based Design	基于组件的设计
CBDMO	Chief Business Development and Marketing Officer	首席商业扩展和营销官
CEO	Chief Executive Officer	首席执行官, 执行总裁
CFO	Chief Financial Officer	首席财务官, 财务总监
CIO	Chief Information Officer	首席信息官
COO	Chief Operating Officer	首席运营官
CPT	Chief Programmer Team	首席程序员队
CSO	Chief Strategy Officer	首席策略官
CTO	Chief Technology Officer	首席技术官
CVS	Concurrent Versions System	并行版本系统, 一种代码版本控制软件

D

DBMS	DataBase Management System	数据库管理系统
DFD	Data Flow Diagram	数据流图
DSS	Decision Support Services	决策支持服务

E

E-business	Electronic Business	电子业务
EIA	Electronic Industries Association	美国电子工业协会

G

GPGPU	General Purpose Computing on Graphics Processing Units	通用计算图形处理器
GPS	Global Positioning System	全球（卫星）定位系统
GUI	Graphical User Interface	图形用户界面

H

HPC	High Performance Computing	高性能计算
HTML	HyperText Markup Language	超文本标记语言

I

IEEE	Institute of Electrical and Electronics Engineers	美国电气和电子工程师协会
IFIPS	International Federation of Information Processing Societies	国际信息处理学会联合会
IoT	Internet of Things	物联网
IPv6	Internet Protocol version 6	因特网协议第 6 版
IS	Information System	信息系统
ITAA	Information Technology Association of America	美国信息技术协会
ITU	International Telecommunication Union	国际电信联盟

L

lb	<拉>libra(e) (= pound(s))	磅
LBS	Location-Based Services	基于位置的服务

N

NATO	North Atlantic Treaty Organization	北大西洋公约组织
------	------------------------------------	----------

O

OO	Object-Oriented	面向对象的
----	-----------------	-------

P

PC	Personal Computer	个人计算机
PDE	Partial Differential Equations	偏微分方程
Perl	Practical extraction and report language	实用报表提取语言

Q

QA	Quality Assurance	质量保证
----	-------------------	------

R

R&D	Research and Development	研究与开发
rep.	representative	代理人, 代表
RFID	Radio Frequency Identification	射频识别技术, 俗称电子标签

S

SCM	Software Configuration Management	软件配置管理
SGML	Standard for General Markup Language	通用标记语言标准
SRS	Software Requirements Specification	软件需求规格说明
SVN	Subversion	一种代码版本管理软件

U

UI	User Interface	用户界面
----	----------------	------

V

V&V	Verification and Validation	验证和确认
VIP	Very Important Person	重要人物, 大人物
VRML	Virtual Reality Modeling [or Markup] Language	虚拟现实建模 (标记) 语言

W

W5HH	Why, What, When, Who, Where, How, How much	
------	---	--

X

XML	eXtensible Markup Language	可扩展标记语言
XP	Extreme Programming	极限编程

Reference of Translation (参考译文)

第一单元 启动软件项目

第一部分 听 与 说

A. 对话：启动软件项目

(Kevin、Sharon 和 Jason 是三名北京航空航天大学软件学院大学二年级的学生。今天，在第四个学期末暑假开始之前，他们正在参加一个班会。)

老师：大家早上好。在这个假期中，大家将要完成一个真正的项目作为你们的课程设计，这里有一些题目，每个人可以根据自己的兴趣和经验来选择，并请在下周之内把你们的决定提交给我。

Kevin：请原谅，老师，请问一下，这个项目是需要每个人单独完成还是几个同学合作完成呢？

老师：我建议团队合作，因为这对于学会未来如何与自己的同事进行合作是有益处的，并且作为一名团队成员，学会怎样互相交流、分享、表达并了解互相的想法也是有益处的。但是每组成员不要超过 4 个人。

Sharon：我感兴趣的题目是四季酒店管理信息系统，你呢，Kevin？

Kevin：哦，这也正是我的想法，我想我们可以组成一个小组。你好，Jason，你愿意加入我们吗？

Jason：哦，好的，我非常愿意！

Sharon：那好，现在我们讨论一下每个人的职责。

Jason：Kevin 善于组织并具有丰富的编程经验，因此我认为他可以担任我们的团队组长或项目经理，负责指导我们的团队和编程实践。

Sharon：我同意。

Kevin：非常感谢你们的信任。好，我一定会尽我所能。关于这个软件，我觉得除了编码以外还需要构建一个数据库并实现一组用户界面。

Jason：我对数据库比较感兴趣，愿意负责建立和管理数据库。

Sharon：我喜欢艺术设计，我认为我可以为项目做用户界面设计和文档撰写的工作。

Kevin: 哦, 不错, 看来这是一个很棒的团队, 它让我感到很有信心! 现在我们根据项目的各个阶段来大致分配一下任务。作为组长, 我来负责需求, Jason 负责设计, Sharon 负责测试。

Jason: 接下来, 我们为项目制订一个粗略的进度计划。

Kevin: 首先, 我们可以根据老师提供的最初的需求文档来设计并完成用户界面操作, 与此同时, Jason 可以构建数据库, 最后我们一起来完成编码。

Sharon: 听起来不错! 但是, 我担心的是, 老师所提供最初的需求文档对我们的设计来说信息可能不全, 所以, 我们必须先根据这个文档来做我们的需求分析, 并完成一个正式的软件需求规格说明书来指导我们的设计。

Kevin: 哦, 是的。多亏你这个重要的提醒。你认为呢, Jason?

Jason: 我完全同意!

(班会结束后, Kevin 从老师那里要了一份关于四季酒店管理信息系统的需求文档)

Kevin: 大家好。我刚从老师那里拿来了酒店管理信息系统的需求文档。

Jason: 让我看看。哦, 有一个关于他们的日常业务的列表, 还有一个相关的需求列表。但是这个看起来似乎有点简略, 没有足够详细的流程。

Kevin: 知道了。这份文档也没有提到酒店的数据流和业务模型。

Sharon: 哦, 这样的话, 我认为我们有必要和客户进行交流以便获取更多的信息。

Kevin: 是的, 这很必要, 我会给客户打电话约个时间。我想在这之前我们也应该做一些事, 最好做一些功课来了解酒店的基本业务和管理。

Sharon: 没错! 非常有必要了解一些相关业务, 这样我们将会充分而准确地理解需求。

Jason: 没问题, 我相信互联网会给我们很多帮助。

B. 听力理解: 软件工程

当今, 计算机软件已经成为以计算机为基础的系统和产品中的关键部分, 并且成为世界舞台上最重要的技术之一。在过去的 50 多年里, 软件已经从解决问题和信息分析的专用工具发展成为业界独立的产业, 并已成为了工业经济的主导因素。

但是, 如何在规定的时间内不超过预算并高质量地开发软件方面, 我们仍然存在问题。为什么完成软件需要如此长的时间? 为什么开发的开销如此巨大? 为什么我们不能在将软件交付客户之前找到所有的错误? 为什么在维护现有程序上要花费如此多的时间和精力? 为什么软件开发和维护过程难以度量? 以上种种问题以及其他问题都显示了业界对软件及软件开发方式的关注, 这种关注导致了业界对软件工程应用的采纳。

那么, 什么是软件? 还有, 软件的哪些特征使其不同于其他人类生产制造的东西? 软件是逻辑而非物理的系统要素, 它的一些特性使其很特别。首先, 客户很难准确完整地给出需求, 并且软件提供者也很难完全理解客户需求。在定义及理解需求的过程中, 特别是需求变更时, 需要不断地进行大量的信息沟通及吸收。其次, 软件看上去容易改变, 其本

质上是无形的，且大部分软件开发过程也是无形的，需要经验、思考及想象；此外，很难完全测试软件。

软件——程序、数据及文档——涉及一个很大范围的技术及应用领域。至今为止，所有的软件还在依照 30 多年不变的一些法则发展进化着。软件工程的目的是为构建更高质量的软件提供框架和解决方案。它包括更加强调系统化开发、集中精力找出客户需求、形式化/半形式化的系统需求规格说明、系统（原型）早期版本的展示，以及更加强调力求保证代码无错等。

从 1968 年“软件工程”一词在 NATO（北大西洋公约组织）会议上首次被使用至今，软件经历了长期的发展。但软件今后的路仍然很长——如果它要和其他工程学科一样成熟。

C. 听写：《人月神话》和《没有银弹》

Frederick P. Brooks 是美国北卡罗来纳大学教堂山分校计算机科学领域的教授。他被称为“IBM/360 大型机之父”，在该系统开发阶段，他担任项目经理；之后在设计阶段，他担任该操作系统软件项目的经理。

他的作品《人月神话》是软件工程中在人为因素方面最经典的著作之一。本书自 1975 年首次出版以来，软件工程师们几乎人手一本。正是在这本书中，Brooks 做出了著名的论断“往进度滞后的软件项目中增加人手，只会使进度更加滞后”，这之后被称作“Brooks 法则”。在本书第一次出版后的 30 年间，软件工具和开发环境可能发生了许多变化，然而在协同工作以及个人与集体本性中的特有非线性规模经济却一点都没有改变。

此外，Brooks 也因其论文《没有银弹》而享誉世界。这篇文章原本是 1986 年发表在 IFIPS 上的论文，在 1987 年又重新发表在 IEEE 计算机杂志上，随后收入《人月神话》第二版中。在此文中，银弹用来比喻使计算机的软件成本像硬件成本那样迅速降低的东西。《没有银弹》的发行量很高，而且引起了热论。它预计，在十年之内，没有任何编程技术可以在软件生产力方面带来数量级的提高。到目前为止，已经过去了九年，Brooks 的预测看上去是准确的。《没有银弹》已经在文献资料领域引发了超过《人月神话》的越来越热烈的讨论。

第二部分 阅读与翻译

A. 软件工程

现在，几乎所有的国家都依赖于以计算机为基础的复杂系统。国家基础设施和公共事业都依赖于以计算机为基础的系统并且大部分电子产品都包含计算机及控制软件。工业制造和配送完全用计算机处理，财政系统也一样。因此，对于国家和国际经济运作，具有成本效益地生产和维护软件是非常重要的。

软件工程是一门工程学科，其研究重点是高质量软件系统开发的经济性。软件是抽象无形的。它不受材料约束、不受物理规律或制造流程限制。在某些方面，这可以简化软件工程，因为软件的潜力不存在任何物理上的限制。然而，这种缺乏自然约束意味着软件可能很容易变得极其复杂，也因此很难被理解。

软件工程的观念是在 1968 年举行的一次讨论什么是后来被称为“软件危机”的会议上首先提出的。这个软件危机是由以集成电路为基础的新计算机硬件的引入而直接导致的。这使得迄今为止不可行的计算机应用成为一个可行的主张。由此产生的软件比以前的软件系统数量更大、更复杂。

开发这些系统的早期经验显示，非正规软件的开发还不够好。大部分项目有时推迟，软件的成本远远超过预期、不可靠，很难维护且运行不稳定。软件开发正处于危机之中，在硬件成本下跌的同时软件成本却在迅速增加。在大型软件系统中，需要有新的技术和方法来控制其固有的复杂性。

这些技术已成为软件工程的一部分，目前正被广泛使用。然而，当我们编制软件的能力增加的同时，所需软件系统的复杂度也随之增加。由计算机与通信系统的一体化以及复杂的图形用户界面而产生的新技术对软件工程师提出了新的要求。因为许多公司仍然不能有效地应用软件工程技术，有太多的项目仍然在生产不可靠、交付逾期和预算超支的软件。

自 1968 年起，我们已经取得了巨大的进展，软件工程的发展显著地改善了我们的软件。我们对软件开发所涉及的活动已经有了更好的理解，并且在软件需求、设计和实施中已经有了成熟有效的方法。新的表示方法和工具减少了开发庞大而复杂的系统所需的工作量。

现在，在软件工程中并不存在单一的“理想方法”。不同类型系统和使用这些系统的不同类型组织的广泛的多样性，意味着我们需要软件开发方法的多样性。但是，过程和系统架构的基本的概念才是所有技术方法的基础，这才是软件工程的本质。

软件工程师确实可以对他们的成就引以为豪。没有复杂的软件，我们就不能探索宇宙，也不会有互联网和现代通信，所有形式的旅行都会更加危险且代价昂贵。软件工程已经做出了很大贡献，随着这门学科的成熟，它会在 21 世纪做出更大的贡献。

B. 云计算

在最近一段时间，云计算变得非常流行并且它涵盖的范围也在持续发展。云计算是一种概念，而不是某种特定的技术。这种概念是指用户通过他们的计算机或手持设备使用的基于互联网的服务器提供的，而不是来自本地设备的应用程序、存储空间以及其他计算资源。

总的来说，云计算指的是一种由“云”服务器进行的计算，这种计算通常通过网络实现。这种类型网络已经投入使用多年，曾经主要用于建立技术研究和低功耗应用所需的超级计算机。在当时情况下，它通常指的是网格运算。现在，云计算指的是通过个人计算机、手机以及其他可以登录互联网络的设备来访问互联网络上的应用程序。云计算的定义为，任何时间、任何地方、任何设备都可以使用的应用程序和数据。举例而言，当在线存储或访问文档、照片、音频以及其他媒体时，当在线使用程序或应用程序（如电子邮件、办公软件、游戏等）时，当在线发布观点、想法以及信息（如社交网络）时，都会用到云计算功能。

尽管当今有很多云应用程序（如 Google Apps、Window Live、Facebook 以及 YouTube）是面向消费者的，其他云应用程序（如计算功能和存储空间、网上购物以及服务器应用程

序)是专门为企业而设计的。企业通常使用公有云上可用的应用程序,但他们也会经常为自己公司的数据创建私有云。

云计算的最大优势主要体现在,由于数据是在线存储而不是本地存储,用户可以在任何互联网覆盖地区访问数据,并且当设备丢失、被盗或损坏时,数据依然安全。此外,在线应用程序通常比需要安装的软件更加便宜。云计算的缺陷主要体现在,当应用程序在云端运行速度低于本地运行速度时,云计算技术可能会降低应用程序的性能,并可能会因公司与个人之间数据传递所需的高带宽造成开销上升。同时,还存在所存储的在线数据被非授权访问或数据丢失等情况的安全隐患。尽管存在这些潜在危险,很多人还是相信云计算会成为新的潮流。

家庭和企业用户选择使用云计算出自多方面原因:

- 可用性:数据和(或)程序可以在全球范围内任何可联网的计算机或设备上使用。
- 节省开支:用户可以避免来自软件和高端硬件(如高速处理器、大容量内存和存储设备)方面的开支。
- 节省空间:用户可以避免来自服务器、存储设备以及其他硬件设备方面的占地空间需求。
- 可扩展性:云计算允许用户根据需要灵活调整计算需求。

云计算允许企业将信息技术基础设施外包,或承包给第三方供应商。企业只需要支付计算功能、存储空间、带宽以及访问应用程序相关的费用。因此,企业不再需要在设备方面进行大规模投资并雇用人员来维护它们。消费者和组织机构通过云计算服务来管理IT基础设施(基础设施即服务),提供应用程序(软件即服务),访问在线数据(数据即服务)并通过基于互联网络的开发平台(平台即服务)来开发应用程序。

第三部分 写作

备忘录

这篇文章将帮助我们解决备忘录写作问题,我们将讨论什么是备忘录,描述备忘录的各个部分,并提供一些例子和解释,使我们的备忘录更有效。

读者和宗旨

备忘录有两方面目的:让人关注问题并解决问题。备忘录通过告知读者新信息(如政策改变、价格上升)或是说服读者采取行动(如参加一个会议或改变目前的生产程序)来达到此目的。无论具体目标如何,当将作者的目的与读者的利益和需求建立起关系时,备忘录就是最有效的。

巧妙地选择备忘录的读者,并确保备忘录发送给的所有人都需要阅读这份备忘录。如果这个备忘录仅仅只涉及一个人,那么就不要再给整个办公室都发送备忘录。另外,确保太敏感的材料不要放在备忘录中;有时候最好的交流方式是面对面交流或者打电话。当发送给少数或适中人数来传达公司或工作目标时,备忘录是最有效的。

备忘录的组成部分

标准的备忘录分为如下几个组成部分来组织信息并帮助作者达到预期目的。

- **标题部分**

标题部分按照如下的通用格式:

收信人: (读者的姓名和称呼)。

发信人: (作者的姓名和称呼)。

时间: (完整的和当前的日期)。

主题: (这个备忘录是有关什么的, 以某种方式重点强调一下)。

请确保按照读者的正确姓名和称呼来发送备忘录, 此外主题务必清晰简洁。

- **开头部分**

通常在开头的第一段包含备忘录的目的、背景和问题、具体的任务或工作。在给读者详细讲解内容和背景之前, 应先让读者对该备忘录有一个大概的了解, 根据备忘录的大纲风格选择特定的序言。备忘录的大纲越直接, 序言就越清晰。备忘录的目的应该阐明读者应该阅读这个文档的原因; 序言部分应该简短, 大约一个简短段落的篇幅就可以了。

- **背景**

背景是作者正在解决问题的事件、环境或背景, 可以用一段或几句话来描述问题的背景及现状, 仅需包括读者所需要的信息, 且清晰明了。

- **任务部分**

任务陈述是备忘录中非常重要的一部分, 在该部分内容中, 应该描述为帮助解决问题作者正在做的事情。在背景中应仅包括决策者所需要的信息, 但要说服读者所存在的问题。不要在无关紧要的细节上漫谈。如果将任务用文字描述有困难, 那么请考虑是否已经阐明了相关情况, 在准备撰写备忘录之前, 作者可能需要做更多的计划。请务必确定使用目的陈述预测将主题分为决策者所需要最重要的标题。

- **摘要部分**

如果备忘录的篇幅长于 1 页, 那么就可能需要一个单独的摘要部分。但是, 对于篇幅简短的备忘录, 摘要部分并不是必需的, 没必要占用大量的空间。该部分为作者所提到的关键建议提供了一个简短陈述, 将有助于读者快速了解备忘录的要点, 该部分也可能包括作者在调查研究过程中所使用的参考资料。

- **讨论部分**

讨论部分是备忘录篇幅最长的一部分, 包含了支持作者观点的所有细节。从最重要的信息开始, 这意味着作者将从关键的结论或建议说起。从最一般的信息开始, 并逐步转移到特定或支持性的事实 (当包含细节时, 请务必使用相同的格式: 从最重要的到最次要的)。讨论部分包括支持作者论点的支持性的观点、事实和调查, 应包括有力的观点和事实依据来说服读者采取作者所建议的行动。如果该部分不充分或不恰当, 那么备忘录将不会像所期望的那么有效。

- **结束语**

在读者读完所有信息之后, 作者应该用一个礼貌的结束语来说明作者希望读者所要采取的行动。作者要考虑到读者将如何从预期的行动中受益, 以及如何可以使这些活动更容易。

- **必需的附件**

请确认在必要时可以通过在备忘录结尾附上列表、图表、表格等附件证明结论或提供

详细的信息。请确认在备忘录中提到附件，并在结束语下方添加关于附件的说明。

格式

备忘录的格式遵循通用的商业写作准则：一份备忘录的长度通常是 1~2 页，并且应该是单倍行距且左对齐。如果要开始新的段落，不要采用缩行，而是要在两段之间空 1 行。商业材料应该简明易读。

作者可以在摘要和讨论部分加上标题来帮助读者更好地理解备忘录，标题应简短而能够阐明该部分的内容。作者所选择的主标题应由开头段落的议题句组成。

为了便于阅读，尽可能将要点或重要的细节列成列表而不是一整段文字，这样更能引起读者的注意，从而帮助读者更好地记住信息。写备忘录时使用列表可以帮助文章更加简洁。

第二单元 获取需求

第一部分 听 与 说

A. 对话：与客户沟通

(Kevin、Sharon 和 Jason 来到四季酒店，会见了酒店业务经理 White 先生及最终用户代表。)

White 先生：欢迎来到我们酒店。

Kevin：谢谢您在您的业务需求文档中提供给我们的关于酒店管理需要的重要信息，但还有一些我们需要弄清的问题。为了能够按照客户的意图完成该系统，我们想向您咨询一些问题。

White 先生：当然可以。

Sharon：White 先生，我正在为四季酒店管理信息系统撰写规格说明书草稿，但我发现可能有一项重要的功能尚不明确。

White 先生：哦，什么问题？

Sharon：假如客户在酒店预订了房间，却没有按时登记入住，这种情况您将如何怎么退还其订金呢？全部、一部分或完全不退？这涉及到如何对退还订金的场景进行描述和建模，以及如何设计这一模块。

Kevin：是的。这对我们在系统中定义一个明确的退还订金机制确实很重要，因为它不仅决定着这个模块本身的设计，还涉及其他模块，如收费的总金额。在确切地定义工作流之后，相应地我们才能知道在系统中如何处理它。

White 先生：好的，我明白了。退还订金确实是房间预订服务中必要考虑的一个事项。现在，我就向你们解释一下遇到这种情况时我们的处理办法。通常，如果一个顾客在距离登记入住时间 24 小时以前或更早的时候取消了他的预订，我们将退还其全部订金；如果在距离登记入住时间 12 小时到 24 小时之间取消预订，将退还其一半订金；但如果在距离登记入住时间 12 小时以内取消，我们将不退还其任何订金。

Kevin: 哦, 好的。除此之外, 退还订金的比率是怎样确定的呢? 在贵宾和普通顾客之间, 或在旺季和淡季之间有什么区别吗?

White 先生: 在我们酒店, 贵宾和普通顾客在价格上没什么区别, 但是我们在不同的季节执行不同的标准。从每年的五月到十月是旺季, 而从每年的十一月到第二年的四月是淡季, 旺季取消预定的时间限制比淡季提前了 12 小时。在旺季, 顾客如果决定退订并想要拿回其全部订金, 就必须在距离登记入住 36 个小时之前取消预订; 如果在距离登记入住前 24 小时到 36 小时之间取消预订, 将退还其一半订金; 如果在 24 小时以内取消, 将没有任何订金退还。

Kevin: 哦, 我明白了。现在看起来更加明确并且更加详细了。我们会在接下来的三天时间内完成需求规格说明书, 然后在本周五用电子邮件发给您。

White 先生: 好, 没问题! 谢谢。

B. 听力理解: 软件需求

在开始设计和构建一个基于计算机的系统之前, 理解需求是十分必要的。要实现这一点, 需要实施一系列需求工程。需求工程发生在与客户沟通和为一般的软件过程所定义的建模活动的过程中, 7 个不同的需求工程功能——起始、获取、精化、协商、规格说明、确认和管理——由软件团队成员实施。

在项目起始阶段, 开发人员和客户及其他项目干系人建立基本的问题需求, 定义首要的项目约束, 并陈述对于系统来说必须要表现出来以满足其目标的主要特色和功能。这些信息在获取阶段得到提炼和延伸, 获取是一项采用辅助会议和用户场景开发的需求收集活动。

精化阶段进一步将需求扩展为分析模型——是一个基于场景、基于活动、基于类、行为的和面向流程的模型元素的集合。可以使用多种建模符号来创建这些元素。模型可能会参考分析模式——在跨越不同的应用程序中已经看到的重复出现的问题域特征。

当需求已被识别并且分析模型已被创建, 软件团队和其他项目干系人协商每项需求的优先级、可用性和相对成本。协商的目的是制定一个现实的项目计划。此外, 每一项需求和分析模型作为一个整体将根据客户需求进行确认, 以确保将要构建的系统是正确的。

C. 听写: 客户与最终用户的不同

软件工程师们与许多不同的项目干系人进行交流, 而客户和最终用户对于接下来的技术工作具有最为重要的影响。在某些情况下, 客户即是最终用户, 但是对于许多项目而言, 客户和最终用户是不同的人, 在不同的企业组织中为不同的管理者工作。

客户是这样一类人或团体:

- (1) 他们最初提出请求要构建软件。
- (2) 定义软件的总体业务目标。
- (3) 提供基本的产品需求。
- (4) 为项目进行资金调配。

在产品或系统企业中, 客户通常是营销部门。在 IT 业中, 客户可能是企业的一个组成

部分或部门。

相反，最终用户则是这样一类人或团体：

- (1) 他们将实际地使用为了达到某种业务目的而构建的软件。
- (2) 将定义软件的操作细节从而使软件的业务目的得以实现。

第二部分 阅读与翻译

A. 软件需求

需求阶段的主要目标是形成一个准确获取客户需求并形成软件开发和确认基础的软件需求规格说明书 (Software Requirements Specification, SRS)。形成软件需求说明困难的主要原因是来自需求参与的三方——客户、最终用户和软件开发人员。需求文档必须能够让客户和用户容易理解，并且能够让开发人员将其作为软件开发的基础来使用。由于在软件需求规格说明中涉及多方人员，存在着沟通上的分歧与隔阂，这使得需求规格说明的任务很难。

需求阶段有三个基本活动。第一个活动是问题或需求分析。该活动的目标是要理解诸如问题的需求、其上下文以及如何适合于客户组织内部等各个方面的问题。第二个活动是需求规格说明，在该活动期间，要对已理解的问题进行详细说明或撰写成文档，产生软件需求规格说明书。第三个活动是需求确认，进行需求确认活动是为了确保软件需求规格说明书中指定的需求正是想要的。

有三种主要方法用于需求分析。非结构化的方法依赖于分析师、客户和用户三者的互动以提出所有需求（然后形成文档）。第二种方法是面向建模的方法，在这种方法中，在可用信息的基础上创建问题模型。模型在确定需求理解是否正确以及确保所有需求是否都已确定方面很有用。建模可以面向功能或面向对象。第三种方法是原型方法，在这种方法中，建立原型来确保需求的正确性和完整性。

要完成目标，软件需求规格说明书必须具有完整性、一致性、无二义性、可证实性和可变更性等特性。一份好的软件需求规格说明书应该详细说明软件需要支持的所有功能、系统性能、存在的设计约束和所有外部接口。

现在流行的用于详细说明功能规格的一种方法是用例方法。使用这种方法，系统的功能通过用例来详细说明，每个用例都详细说明当一个用户为了完成某个目标而与系统交互时的系统行为。每个用例既包含正常情况，也包含异常情况，因此可以提供系统行为的完整描述。尽管用例是为规格说明而设定的，但由于其自然性和故事性，通过在不同的抽象级别上表达用例，用例也可以用于问题分析。

对于需求确认，最常用的办法是评审和检查需求。在需求检查中，评审小组中也应该包含一名客户代表，以确保获取了全部需求。

B. 移动互联网

移动互联网 (mobile Web) 指的是使用诸如智能手机、功能手机等手持移动设备，通过连接移动网络或其他无线网络来接入万维网，比如基于浏览器的互联网服务。

从传统的角度讲，使用万维网就是在大屏幕笔记本电脑和台式机上通过固定电话服务

连接网络。然而,现在的万维网变得更加易于通过便携设备和无线设备连接。2010年的国际电信联盟(ITU)报告说,以当前的增长速度,通过笔记本电脑和移动设备登录万维网的用户数量可能会在5年内超过使用台式机登录的人数。2007年以来多点触控智能手机和2010年以来多点触控平板电脑的兴起,加速了向移动互联网接入的转变。这两种平台都提供了比上一代移动设备更好的互联网连接、屏幕、移动浏览器或基本应用程序的用户互联网体验。网站设计者可能会单独制作这些页面,或让这些页面可以自动转化,如维基百科移动版。

移动互联网应用(移动应用)和本地应用的区别正在变得模糊,因为移动浏览器可以直接访问移动设备(包括加速计和GPS芯片)的硬件,因此基于浏览器的应用的速度和功能都得到提升。对于高端的用户界面图形功能的长期存储和应用会进一步减少开发特定平台本地应用的需求。

时至今日,移动互联网仍然受到互用性和可用性问题的困扰。互用性的问题源自移动设备、移动操作系统和浏览器平台的多样性。可用性的问题集中于手机的有限尺寸(对分辨率和输入操作的限制)。尽管存在这些问题,很多移动开发方还是编制出使用移动互联网的应用程序。如2011年6月份的一个有关移动开发的研究所显示的那样,移动互联网是继Android和iOS平台后的第三大最多使用的平台。

根据美国计算机协会(ACM)2013年4月的一篇文章,互联网技术专家Nicholas C. Zakas指出,人们在当下使用的手机已经比1969年7月用于登月的阿波罗11号的70磅(32公斤)阿波罗导航计算机更加强大。尽管如此,移动设备依然受到低网速影响,这点与1996年的情景类似。移动设备更低的下载速度/反应时间、数据传输延迟,以及“高延迟连接、更低速的CPU和更少的内存”,迫使开发者重新考虑原本为台式机设计的“有线连接、高速CPU和大量内存”的互联网应用。

“移动网络”(mobile Internet)指的是通过蜂窝电话服务供应商来连接互联网。这种无线链接可以在用户切换服务区的时候自动更换信号塔。它可以指代持续连接同一个信号塔的固定设备,因此“移动网络”不一定真的“移动”。通常来讲,不移动的用户可以使用Wi-Fi或其他更好的方式。相比之下,通过电话系统的蜂窝基站比直接连接互联网服务供应商的无线基站所提供的服务更加昂贵。

诸如智能手机等不通过蜂窝基站的方式连接数据或语音服务的移动电话,不使用移动网络。配有带宽调制解调器和蜂窝服务供应商服务的笔记本电脑,则在移动网络上传输信息。

移动带宽调制解调器可将智能手机“绑定”在一个或多个计算机或最终用户设备上,通过蜂窝电话服务提供商所能提供的协议,从而提供对互联网的连接。

第三部分 模拟写作

软件需求规格说明书

需求规格说明书涵盖了与需求定义完全相同的范围,但这是从开发者的角度来看的。凡是以客户的语言所描述的需求定义,用来指向客户世界里的对象、状态、事件和活动,

在需求规格说明书中都应按照系统界面进行描述。我们通过重写需求来完成这项工作，以便使需求仅仅指向那些现实世界中对于拟议系统能感知的或是能被此系统所驱动的对象（状态、事件、活动）：

1. 在记录系统的接口时，我们要详细描述所有的输入和输出，包括输入源、输出位置、输入和输出数据的取值范围和格式、用来控制特定的输入输出转换顺序的协议、窗口的格式和组织以及任何时间约束。请注意，用户界面通常不是唯一的系统接口；系统可能还会与其他软件组件（如数据库）专用硬件、因特网等进行交互。

2. 接下来，我们要根据接口的输入输出重申所需要的功能。我们可以使用功能符号或数据流图将输入到输出绘成路线图，或使用逻辑来描述功能的前置条件和后置条件。我们可以使用状态机或事件路线来说明操作的准确顺序或输入输出的准确排序。我们可以使用实体-关系图把相关的活动 and 操作放入类中。最后，规格说明书应当是完整的，也就是说它应该为任何可行的输入顺序指定了输出。因此，要包括对输入的合法性检查和对异常情况的系统响应，如不合法的前置条件。

3. 最后，我们针对每个客户的质量要求制定适合的标准，以便可以最后论证系统是否符合这些质量要求。

作为最终结果的文档是一份对开发者将要开发的产品的描述，应当提供充分详细的细节以区分可接受和不可接受的解决方案，但不需要说明拟议系统应当如何设计和实现。

第三单元 项目计划

第一部分 听 与 说

A. 对话：软件项目计划

Jason: 这么多工作要做啊！

Kevin: 我认为我们需要一个正式的项目计划作为指导。

Sharon: 是的，首先是时间，这也是影响我们项目最重要的因素之一，我们需要一个进度表，尤其是项目的最终期限。

Kevin: 我们总共有 40 天的时间，而需求已经花了 5 天的时间，因此我们还有 35 天时间。

Jason: 哦，听起来时间确实很紧张。

Sharon: 我们似乎应该尽快开始编程，对吗？

Jason: 尽管编程是软件项目中非常重要的一个部分，但是，我认为现在最重要的事是，必须确定一个合适的时间进度表来控制我们的进度，并确保能够按时交付。

Kevin: 是的，在需求阶段，我们还需要 3 天时间来对需求进行描述、分析和建模。之后，我们需要 3 天时间来完成体系结构设计，5 天时间完成详细设计；由于这些工作量都是用于软件设计，因此接下来的编程相对不会很难，我认为在一周内应该能完成；测试及后续的调试大约 10 天完成。

Sharon: 也许我们可以用甘特图来描述进度表, 对于项目计划来说, 甘特图是一种形象生动并且实用的工具。

Kevin: 好主意!

Jason: 事实上, 测试不应该被看成是仅在编码阶段完成后才开始并仅限于以检测错误为目的。实际上, 测试计划在需求阶段的早期就应该开始了, 并且测试计划和流程必须系统、不断地制定, 并且随着开发的不断展开, 尽可能地改进。我觉得, 在编码阶段, 我们可以同时进行单元测试, 这将会节省大量时间并会获得更好的测试效果。最后, 我们可以通过与客户一起工作来进行验证测试, 检查开发出的软件客户是否满意, 以确保客户能够获得他们所需要的。

Sharon: 顺便提一下, 我们需要三台计算机, 必须安装客户所要求的具有统一版本的软件作为开发工具, 以微软公司的 Visual Studio 2012 作为开发平台, 以微软公司的 SQL Server 2012 作为数据库管理系统。

Jason: 我们还需要网络。

Kevin: 我将尽快为大家准备开发环境, 然后给每个人分配一些管理任务。Sharon 负责文档管理, Jason 负责变更管理, 我负责软件质量保证, 怎么样?

Jason 和 Sharon: 没问题。

Kevin: 好的, 我已经把刚才讨论的内容做了记录, 并会在两天内完成一份项目计划文档。

B. 听力理解: 软件项目计划

关于计划有许多种不同的观点。有些人是“最简单派”, 他们认为, 变化经常使得进行详细计划变得没有必要。其他人是“传统派”, 他们认为, 计划提供了有效的路线图, 并且计划越详细, 团队失败的可能性就越小。还有一部分人是“敏捷主义者”, 他们认为, 一个快速的“计划活动”是有必要的, 但路线图将会在软件开始时作为实际工作出现。

那么到底应该如何做? 在许多项目中, 往往会因过度计划而事倍功半(太多的事情在变化), 但过于简略的计划也会导致开发混乱。像生活中的大多数事物一样, 应该适度进行计划, 足以为团队提供有用的指导即可, 不用多也不用少。

在一篇优秀的关于软件过程与项目的论文中, Barry Boehm 提出: 需要按比例缩减的组织原则来为简单的项目提供简单的计划。Boehm 建议一种方法来处理项目目标、里程碑、日程表、项目分工职责、管理及技术手段以及所需资源。他称此方法为 W5HH 原则, 在一系列引出关键项目特性的定义和随之产生的项目计划的问题之后: “为什么要开发系统?”、“要做什么?”、“什么时候完成?”, 上述三项是最必不可少的。此外, 还有“一个功能由谁负责?”、“组织机构位于哪里?”、“工作如何在技术和管理上完成?”以及“每项资源各需要多少?”。对于 Boehm 的 W5HH 问题的回答十分重要, 这点与软件项目的复杂度及规模无关。

C. 听写: 软件项目中的四个变量

项目是使用资源来达到预定目的和目标的一系列精心确定的活动, 是具有特定起止日期的有限努力和具有一系列的时间安排、预算及组织约束条件的管理过程。

通常认为,在软件项目中,我们将控制四个经典变量——费用、时间、范围和质量。这四个变量通过相互作用共同影响着整个项目:增加的范围往往意味着要花费更多的时间和可能超过预算;紧迫的时间压力要求更强的资金支持,同时质量也可能受到影响;资金不足就难以解决范围之内的客户业务问题;更高的质量可能需要更长的开发生命周期和更高的费用。

看起来这四者之间的关系并不那么简单。例如,不能仅仅依靠投入更多的资金提高软件的开发速度。正像谚语所说:“即使是九个女人,也不可能在一个月之内生出孩子。”

Kent Beck 在他的《解析极限编程:拥抱变化》一书中提到,解决方法就是使这四个变量是可见的。如果每个人——程序员、客户和经理——都能注意到这四个变量,那么他们就可以有意识地选择控制哪些变量。如果他们不喜欢第四个变量可能带来的结果,那么可以改变投入或是选择不同的三个变量来进行控制以最终达到项目目标。

第二部分 阅读与翻译

A. 软件项目计划

一份恰当的项目计划是一个项目成功的重要因素。没有合适的计划,软件开发项目是不可能成功的。项目需求和架构可用之后就可以做一个好的计划了。项目计划的重要活动包括过程计划、工作量估算、进度和人员安排计划、质量计划、配置管理计划、项目监控计划和风险管理。

过程计划一般涉及选择一个合适的过程模型并对它进行剪裁,以适合项目需求。在工作量估算中,项目总体工作量需求和不同阶段的工作量分解都要估算。在自上而下方法中,首先估算总体工作量,经常从项目规模估算开始,然后再确定不同阶段或任务的工作量。在自底向上方法中,首先识别项目中的主要任务并估算其所需要的工作量。从任务的工作量估算开始,进而完成总体工作量估算。

项目的总体进度和主要里程碑依赖于工作量估算和项目人员水平,并且可以使用简单模型来从工作量得到一个粗略的进度估算。很多时候,一个整体进度通过使用模型来确定,然后进行调整以满足项目需求和约束。详细进度是指在保持总体进度和工作量估算的条件下,将任务分解为更小的、可安排的任务,然后将其分派给特定的团队成员。详细进度是最生动的项目计划文档,因为它列出了必须做的任务;项目计划中的任何变化都必须适宜地反映在详细进度中。

质量计划对于确保最终产品的高质量非常重要。项目质量计划确定了开发的不同阶段中所有必须执行的验证和确认活动以及如何执行这些活动。

配置管理的目标是要控制项目过程中发生的变化。配置管理计划确定了需要被控制的配置项,并指定了完成配置计划的程序以及控制如何访问。

风险是那些不一定会发生、但一旦发生就会对项目具有负面影响的事件。为了即使在风险存在的情况下能够实现项目目标,需要适当的风险管理。风险管理要求对风险进行识别、分析和优先级排序。然后制订和执行风险缓解计划以将最高优先级风险的影响降到最小。

为了使项目计划成功实施,至关重要的一点是谨慎地监控项目。活动级别的监控、状态报告和里程碑分析是常用的机制。对于分析和报告,实际工作量、进度、缺陷和规模应

被度量。有了这些度量,才有可能依照计划监控项目的执行情况。在此监控的基础上,如果需要,便可以采取措施来纠正执行的过程。

总体而言,项目计划制订了为了达到项目目标所应遵循的路线。项目计划指定了项目成员应当执行的所有任务,并指定了为执行该计划谁做什么、用多少时间以及什么时候做。有了详细计划,余下将要做的就是执行计划。当然,由于工作不能总是按计划完成,因此计划从来不是一成不变的。通过适当的监控,这些情况能够被识别,从而可以相应地改变计划。基本的项目计划原则和技术也可以用于计划修改。

B. 大数据

在近五年以前,各种企业收集的大多数数据都是结构化的交互数据。这些数据可以简单的填充到关系数据库管理系统的行、列当中。从那时以来,来自网站流量、邮件信息以及社交媒体(微博、状态信息)甚至音乐播放列表以及传感器数据带来了数据爆炸。这些数据的无结构或半结构化使得它们不适用于通过行、列管理数据的关系数据库产品。流行词“大数据”指的就是这种蜂拥而至的大部分来自互联网并且源源不断流向全世界机构的数据。由于这些数据的容量过于庞大,传统的数据库管理系统无法在可接受的时间内捕获、存储和分析它们。这些“大数据”带来的挑战如,通过分析每天产生的 12 太字节(10^{12}B)的微博数据来更好地了解用户对产品的看法;分析 1 亿封电子邮件以便在邮件信息旁边投放合适的广告;或分析 5 亿条详细的呼叫记录来寻找诈骗和粗制滥造的方式。“大数据”的概念以及用于处理它的工具由 Google 等搜索引擎最先提出。Google 所面临的问题是:它需要每天处理 5 亿条搜索,并且在几毫秒内显示搜索结果和相应广告。可以试试搜索词条 big data,你会发现 Google 在 38 毫秒(约三分之一秒)的时间内回复了超过 10 亿条结果。这个速度比人们阅读句子的速度还要快!

“大数据”通常指的是拍字节(10^{15}B)和艾字节(10^{24}B)规模的数据,数十亿到数万亿的,来自不同领域的记录。换句话说大数据产生的数量和速度远远大于传统数据。尽管每条 Twitter 文本限制在 140 个字符以内,它每天仍然产生了超过 8TB 的数据。根据 IDC 技术研究公司的调研,数据每两年都会增长一倍以上,因此对企业而言所需面对数据量不断激增。了解数据所代表的信息对于占据市场优势而言至关重要。

企业对“大数据”感兴趣,是因为它们与更小的数据集相比包含了更多的模式与有趣的特例,因而具备提供新的针对顾客行为、天气模式、金融市场活动或其他现象相关信息的潜力。然而,为了从这些数据中获取商业价值,企业需要能够对自己传统企业数据以及非传统数据进行管理和分析的新技术和新工具。

为了管理这些大量的无结构、半结构化的数据,以及结构化的数据,企业通常使用 Hadoop 框架。Hadoop 是一个由 Apache 软件基金会管理的开源软件框架,可以在廉价的计算机上分布并行处理大量的数据;可以将大数据问题分解为多个子问题,然后将它们分布在数千个廉价的计算机处理节点上,并将分析结果组合在一个较小、更易于分析的数据集上。你或许就曾经使用 Hadoop 在网络上寻找最优惠的飞机票价、查询某家餐厅的位置或通过 Facebook 与你的朋友保持联系。

Hadoop 可以处理大量任何种类的数据,包括结构化的交互数据、松散结构的数据如 Facebook 和 Twitter 的订制信息、类似网络服务器日志文件的复杂数据以及无结构的音频、

视频数据。Hadoop 运行在一群廉价的服务器上,可以根据需要添加和删除处理器。公司可以使用 Hadoop 来分析大量数据,以及在填入数据仓库前将无结构和半结构数据进行转化。Facebook 就在它巨大的 Hadoop 集群上存储了大量数据,这些数据大约有 100 拍字节(10^{17}B)。这个数量是美国国会图书馆的 1 万倍。Yahoo 公司使用 Hadoop 来跟踪用户的行为,从而调整它的主页,以满足用户需求。生命科学研究公司 NexyBio 使用 Hadoop 以及 HBase 来处理制药公司染色体组研究的数据。顶级数据库供应商,如 IBM、Hewlett-Packard、Oracle 以及 Microsoft 等公司都有他们自己的 Hadoop 分布式软件。其他供应商也提供将数据从 Hadoop 中移入、移出或是在 Hadoop 中分析数据的工具。

第三部分 模拟写作

软件项目计划

为了与客户交流风险分析与管理、项目成本估算、进度及团队组织,我们通常需要撰写被称为项目计划的文档。该计划以书面形式说明客户的需求,以及为了满足该需求我们希望采取的行动。客户可以通过该计划了解开发过程中活动的有关信息,从而在开发过程中很容易地跟进该项目的进展。我们也可以使用该计划与客户确认所作的假定,尤其是关于成本和进度。

一个好的项目计划应包括以下内容:

1. 项目范围。
2. 项目进度。
3. 项目团队组织。
4. 被提议系统的技术描述。
5. 项目的标准、程序、建议的技术和工具。
6. 质量保证计划。
7. 配置管理计划。
8. 文档计划。
9. 数据管理计划。
10. 资源管理计划。
11. 测试计划。
12. 培训计划。
13. 安全计划。
14. 风险管理计划。
15. 维护计划。

范围规定了系统边界,对系统将包含和不包含的内容做出解释。它向客户保证我们理解客户想要的东西。进度可以使用工作分解结构、可交付物和活动时间表来描述,以显示在项目生命周期过程中的每一个点上将会发生的事情。在阐明解释某些开发任务的并行特征时,甘特图是很有用的。

项目计划还列出了开发团队的人员、组织方式及任务。正如我们所见,在整个项目中,并不是每个人在项目过程中随时都是必需的,所以项目计划通常包含一个资源分配图,用

以显示在不同时间人员的配置。

撰写技术描述促使我们在预期开发将如何进展时去回答问题和解决问题。该描述列出了硬件和软件,包括编译器、接口和专门用途的设备或软件。任何关于布线、执行时间、响应时间、安全或其他功能或性能方面的特殊限制均应撰写在项目计划中。

第四单元 团队合作

第一部分 听与说

A. 对话: 团队结构

Jason: 考虑到我们目前的情况,我们可以在开发过程中尝试一下极限编程。极限编程主张没有任何障碍的沟通并实现快速的交付,这对我们团队和项目的特点来说非常适合。

Sharon: 没错。使用极限编程,我们能够互相合作并集中我们的时间和精力,这正是成功所必需的。

Kevin: 据我所知,还有另外一种团队结构,在这种团队结构中,软件开发过程中有三种主要的任务类别——与管理相关的、与开发相关的和与测试相关的。使用这种方法,我们可以清晰地将工作划分为上述的三个主要方面: Jason 主要集中于开发, Sharon 进行测试,而我负责项目管理,这样大家就可以更加专注地开展工作了。

Sharon: 听起来不错!那我们就开始吧!

Jason: 另外,在日常工作中,开发团队成员之间良好的沟通非常重要。我们必须就各自的工作情况、所做出的设计决定和对先前必要的决定的变更进行交流。

Sharon: 要做到这一点,我们最好在彼此之间建立一些正式和非正式的沟通机制。正式的沟通可以通过撰写文件和进度报告来进行。另外,我们可以通过因特网随时非正式地交流想法。

Kevin: 此外,由于我们将一起同步合作开发,因此需要一套配置管理机制以确保大家工作的一致性。

Jason: 是的,或许一些自动化工具可以帮助我们。

Sharon: CVS,是并行版本系统的缩写,是业界使用最广泛的版本控制系统,而且非常适合像我们这样规模和特点的开发团队。我们可以尝试一下。

Kevin: 好的,我同意。

B. 听力理解: 项目团队

软件项目会因诸多原因陷入困境。许多项目开发工作量很大,需要两人到几百人规模不等的项目团队开发。这会导致在协调团队成员方面的复杂、混淆和巨大困难。因此,将一个团队聚在一起高效率地工作是一项关键的管理任务。一个好的团队具有团队精神,其中的成员在被自己的个人目标所激励的同时,也会被团队的成功所激励。

Ian Sommerville 在他的《软件工程》一书中将这种精神描述为“聚合性”。在一个内聚性团队中，团队成员会把团队利益置于个人利益之上。一个具有好领导、内聚性的团队中成员对这个团队会很忠诚。他们认同团队目标并将其他成员视为一体。他们会尽力保护团队，作为一个集体，免受外部干扰。这能够使得团队变得健壮，并且能够处理问题和意外状况。这样的团队可以通过提供彼此之间的支持与帮助来处理变化。内聚性团队的好处包括全体一致建立并由成员自我遵守的团队质量标准，一起紧密工作、相互学习、逐渐了解各自的工作、实践无自我编程等。

在另一本书《人件》中，DeMarco 和 Lister 把这种精神称作“黏合”。他们论述该问题如下：一个“黏合”的团队是一群非常紧密联合的人，以至于他们的集体能量大于个体能量之和。一旦团队开始“黏合”，成功的概率就会增加，这样的团队可以变得不可阻挡，成为一股不惜牺牲或成功的力量……他们不需要用传统的方式进行管理，并且当然不需要被激励，就已经劲头十足。

为了加强团队的聚合性及黏合性，必须采取一些有效的手段，以便协调成员使得其了解团队其他成员的动力、优点及弱点。在软件开发组成员间进行良好的沟通是最基本的方式之一。团队成员必须就他们的工作状态、已做好的设计决策以及针对以前决策所作的必要改动交换信息。如要完成这个目标，必须建立组员之间一些正式及非正式的沟通机制。正式沟通通过“书面、安排的会议以及其他相对非互动的、非个人化的沟通渠道”来实现，非正式的沟通则更加个人化。软件团队的成员可以每天分享彼此的经验、寻求帮助、彼此互动。

C. 听写：敏捷软件开发

产生基于计算机的系统及软件产品的现代商业环境步伐很快且不断更新。市场的情况在迅速变化，最终用户的需要在不断进化，新的竞争威胁毫无警示地出现。许多情况下，在项目开始前，我们难以充分地定义需求。软件工程师必须足够敏捷，才能适应不断变化的商业环境。

敏捷软件开发的最新定义是在 20 世纪 90 年代中期逐渐发展形成的。起初，敏捷开发方法被称作“轻量级方法”，作为对比“重量级方法”的一部分。这种方法的目标是用一种更轻便、更迅速并且以人为中心的方式开发软件。2001 年，这种“轻量级开发方法”的一些非常著名的支持者成立了“敏捷联盟”并签署了“敏捷软件开发宣言”，这是一个由支持者们共享的价值观的陈述，针对那些正在关注新的敏捷开发过程的人，那就是：个人和交互胜过过程和工具，可以工作的软件胜过面面俱到的文档，客户合作胜过合同谈判，应对变化胜过遵循计划。

实际上，许多类似敏捷开发的方法早在 2000 年之前就已经出现了，极限编程（简称 XP）就是 Kent Beck 于 1996 年所创建的一种应用非常广泛的敏捷过程。Edmonds 于 1974 年在其论文中介绍了一种自适应软件开发过程。其他的著名的敏捷开发方法还有 Scrum、Crystal、自适应软件开发（ASD）、特性驱动的开发（FDD）以及动态系统开发方法（DSDM）等。

第二部分 阅读与翻译

A. 团队结构

我们已经看到，当进度规划好之后，资源的数量就已经确定。详细进度安排只有在实际的人员分配已经完成之后才进行，因为任务分配需要团队成员能力的相关信息。一般而

言, 我们都会假定该项目团队由一个项目经理领导, 由项目经理来进行规划和任务分配。这种分层次的团队组织形式相当普遍, 早期被称作首席程序员团队。

在这种层次型组织中, 项目经理负责该项目的重大技术决定。他负责大部分的设计并将设计的不同部分的编码工作分配给程序员。团队通常由程序员、测试员、配置管理员构成, 可能还有一名文档管理员。可能还有其他的角色, 像数据库管理员、网络管理员、后备项目经理或后备配置管理员。应该指出的是, 这些都是逻辑上的角色, 一个人有可能会担任多个角色。

对于一个小项目, 一个级别的层次就足够了。对于较大的项目, 通过将项目划分成模块, 并且拥有模块领导, 他们负责与其模块相关的所有任务并带领一支执行这些任务的团队, 这样的组织能够很容易地扩展。

另一种与之不同的团队组织是无自我团队。无自我团队由 10 个或更少的程序员组成。小组目标由全体成员的一致同意来设定, 并且来自每位成员的提议可用于重要决定。团队领导由团队成员轮流担任。由于其这样的性质, 无自我团队有时被称为民主团队。这种结构允许所有成员的投入, 从而可以为难题找出更好的决策。这种结构非常适合于没有时间限制的长期研究型项目。但它并不适合于有时间限制的经常性的任务, 对于这样的任务, 民主结构中的交流是不必要的并且会导致效率低下。

最近, 针对非常大型产品的开发, 另一种结构已经出现。这种结构认为, 在软件开发中有三种主要任务分类——管理相关、开发相关以及测试相关的任务。这种结构还认为, 让测试和开发团队相对独立, 以及不让开发人员或测试人员向非技术性经理报告, 通常可以收到令人满意的效果。因此, 在这种结构中, 有一个总体的单位经理, 在他之下有三个小的层次组织——针对项目管理、针对开发以及针对测试。开发者的首要工作是编写代码并在开发经理的领导下工作。测试者的责任是测试代码并在测试经理的领导下工作。项目经理为正在构建的软件提供规格说明, 并确保开发和测试得到妥善协调。在一个大型产品中, 这种结构可以在每一个主要的单元中复制。这种类型的团队组织已被应用于像微软这样的公司。

B. 物联网

讨论互联网的未来却不提及物联网 (IoT) 是不完整的, 物联网有时也被称为工业网络。

互联网技术广泛应用在台式计算机、笔记本、平板电脑和智能手机上, 在民用电子、电器、汽车、医疗设备、公共事业系统、各种类型的机械装置, 甚至衣服上——几乎所有可以装备传感器的东西, 来收集信息、连接互联网, 并且用数据分析软件进行数据分析。物联网建立在现有技术的基础上, 如射频识别技术 (RFID)。随着低成本传感器的可用性提高, 数据存储费用的降低, 可以处理万亿数据的“大数据”分析软件的发展, 以及允许为所有这些新设备分配互联网地址的 IPv6 技术的实现, 物联网逐渐成为可能。对于物联网的研究和资助, 主要由欧盟、中国 (被认为是遥感地球) 以及美国的公司 (如 IBM 的智慧地球计划) 来领头。尽管在物联网完全实现之前, 挑战还是存在的, 但现在已经越来越接近成果了。如今, 物联网还在最初阶段, 主要用于远距离监视事物。

例如:

- 农场主通过牛身上的无线传感器来得到牛生病或走失的警告;
- 穿戴式医疗技术允许医生监测病人的慢性疾病;

- 家用电器上的传感器在当器械需要维护或修理时可以提醒制造商。

最终，物联网的功能将不仅仅局限于告诉你物体的位置和情况。设备之间将会相互交流并且实现自治。这被称为机器对机器（M2M）的技术。

例如：

- 汽车可以自动驾驶；
- 交通灯会自动响应交通拥堵和交通事故等情况，以引导车辆远离路况糟糕的区域；
- 房屋会在房间中无人的情况下，自动关闭灯光和加热系统。

最终，提供的每一件商品都会被追踪并且智能化。

同事的行踪也同样可随时掌握。INEX 咨询顾问 W. David Stephenson 说“每一个级别的雇员都会因此变得更加高效”。这将会使得经理们运行公司的模式发生转变，摆脱分层决策者（老板的经典角色）。Stephenson 表示，“在我眼里，这将是物联网带来的最大益处”。

尽管物联网听起来很酷，但它也存在缺点。其隐私性、可靠性以及数据控制方面的问题仍然亟待解决。

但即使如此，现在没有什么能够阻止物联网的发展。在几年之内，它会像手机和宽带一样，成为人们生活的重要组成部分。

第三部分 模拟写作

PowerPoint 演示文稿

简介

PowerPoint（PPT）演示文稿和海报演示很类似，唯一不同的是 PPT 是在计算机上用幻灯片来演示，而不是用真正的海报。这种形式往往用来配合口头报告，可以使报告更具说服力。我们还可以加入声音和视觉媒体。幻灯片演示通常用于与一个大的群体分享信息，如专业会议、课堂演示和会议等。幻灯片演示应该更像是报告的提纲。在幻灯片演示中有三个主要元素：文本、图像、表格或图表。文本可以突出主要观点以及关键术语和概念；图片可以说明和强调主要观点；表格和图表可以通过一种容易理解和观看的方式展示信息。

一些准备工作

在开始做 PPT 之前，我们应该先考虑清楚演示的几个关键部分：观众、目的（说服力、增长知识性等）、主题以及陈述。因为一个好的 PPT 设计应该适合于演示的场合，因此在开始制作 PPT 之前，尽量搞清使用何种措辞有助于成功的演示。

主要组成部分

PPT 的幻灯片应该更像一个提纲。文字通常都是将主要内容列出来，而不是完整的句子。以下是几个可以在幻灯片中展示的内容：

- 图表和/或表格；
- 解释；
- 列表；
- 基本事实；
- 必要的图片。

内容安排

PPT 内容的顺序应该取决于观众所需。无论做什么，都要谨慎地组织 PPT 并井井有条地陈述论点，这样观众才会信服我们的论点。以下是一些可以参照的格式。

- 概述、主体、结论（此类格式通常用于帮助观众弄清我们的演示，否则观众可能会被复杂的论点搞糊涂）；
- 轶事、内容、结论（此类格式主要为了防止观众在开始演示之前就已经厌烦）；
- 计划、好处、轶事（此类格式主要用于介绍一些新鲜事物，而且所面对的观众要求我们的演示简短且切题）。

其他有用的信息

提倡的内容：

- 为整个 PPT 选择一个统一背景；
- 使用简单清晰的字体；
- 保证字体足够大，使后排的观众也能看到；
- 在列表中使用项目符号和一致的短语结构；
- 列出基本信息即可，运用关键词来引导读者或观众贯穿整个演示过程；
- 使用直接、简洁的语言，控制文字数量到最少；
- 必要时提供相关的解释；
- 使用空格来隔开文本或图片表格之类的可视组件；
- 保证每张幻灯片之间的联系符合逻辑；
- 为每张幻灯片加上标题。

避免的内容：

- 在幻灯片中堆满了图表；
- 使用复杂的字体；
- 添加不相干的信息；
- 写下所有要说的话；
- 使用分散观众注意力的图像；
- 颜色搭配难以阅读，如在蓝色背景上使用黑色的字体，尽量使用对比度高的组合。

第五单元 系统设计

第一部分 听 与 说

A. 对话：软件设计

Sharon: 系统分析建模和需求规格说明已经完成，我们可以进行下一步啦！

Kevin: 是的，现在我们讨论设计过程中的一般任务集。

Jason: 我们将从软件的一个“宏观”视图逐步转移到定义了系统实现所需细节的微观视图。
我认为，这个过程应该从关注于体系结构开始。

Sharon: 你的意思是？

Jason: 使用上周构建的分析模型, 我们需要选择一种适合我们软件的体系结构风格。

Kevin: 然后, 我们可以把分析模型划分成设计子系统, 并将这些子系统分配到体系结构中。

Jason: 在详细设计阶段, 我们必须确定每个子系统在功能上都是关联的, 并首先设计子系统接口, 然后将分析类或功能分配给每个子系统。

Kevin: 此外, 拥有用户欣赏的用户界面对于软件尤其是对于像我们这样商业软件的成功来说是一个重要的因素。

Jason: Sharon 是一个称职的美术设计师。

Sharon: 我会尽力的!

Kevin: 我们将一起合作在用户场景的基础上详细说明行为顺序, 并帮助 Sharon 来创建界面的行为模型。

Sharon: 好的, 接下来, 我将定义界面对象和控件机制, 必要时还拜托你们进行评审和修订。

Kevin: 没问题。在完成体系结构设计之后, Jason 和我将通过在较低的抽象级别上详细说明所有算法来进行组件级设计, 定义组件级的数据结构, 有必要还可以完善每个组件的接口。

Jason: 组件级设计评审之后, 我们就可以为系统制定部署模型了。

B. 听力理解: 软件设计

设计工程开始于需求工程的首次迭代结束时。软件设计的目的是应用一系列能够引导高质量系统或产品开发的原则、概念和惯例。设计的目标是创建一个软件模型, 该模型将正确地实现所有的客户需求并为那些使用该软件的人带来快乐。设计工程师必须从大量可供选择的设计方案中筛选并最终集中形成一个最切合项目干系人需要的解决方案。

设计过程从软件的“宏观”视图向更微观的视图转移, 后者定义了系统实现所必须的细节。设计过程开始于关注体系结构设计。然后定义子系统, 建立子系统之间的通信机制; 确定构件, 撰写每个构件的详细描述。另外, 还要设计外部接口、内部接口和用户接口。

设计概念在软件工程发展的前半世纪中不断进化。设计概念描述了那些与所选择的软件工程过程、所应用的设计方法或所使用的编程语言都无关计算机软件的属性。

设计模型包含四种不同的元素。随着每个元素的开发, 更完整的设计视图逐渐形成。体系结构元素使用来自于应用域、分析模型以及模式与风格的可用目录的各种信息, 以获得软件、其子系统和组件的完整的结构表示。接口设计元素对外部和内部接口及用户界面进行建模。组件级元素定义构成体系结构的每个模块(组件)。最后, 部署级设计元素将体系结构、其组件和接口分配到将要安装软件的物理结构中。

C. 听写: 用户界面设计

当界面的概念首次出现时, 它通常被理解为人和计算机之间进行交流所借助的硬件和软件。因此, 用户界面设计也被称为人机交互界面, 简称为 HCI。随着这一概念的不断进

化,现在也包括用户体验的认知和情感方面的内容。

许多技术创新都要依靠用户界面设计来将它们的技术复杂性提升为可用的产品。技术本身可能并不能赢得客户的认可和市场。用户体验,或者说用户对最终产品的体验才是认可的关键。也正因为如此,用户界面设计成为了设计过程的一部分。产品的设计师主要关注技术层面,而可用性专家则关注用户界面。为了达到最高的工作效率且具有高性价比,这种工作关系应该从项目开始一直保持到产品首次推出。

用户界面影响着用户的感受、情绪和心情。如果用户界面有错误,用户将会感觉他们无法控制软件,自然而然就会气恼并指责软件。相反地,如果用户界面很漂亮并且按照用户期望的方式工作,用户将会因为通过操作完成目标而非常高兴。因此,所有用户界面设计的最重要原则是,当程序按照用户所认为的方式正确地运转,就可以认为用户界面设计得很好。

为了使用户满意,应该使用户能够完全控制他们的工作环境。要做到这一点,需要正确解释用户的行为。此外,还应该减轻用户的记忆负担,并使界面风格一致。上述三点被认为是用户界面设计的“黄金原则”,而实际上也形成了一系列用户界面设计原则的基础,这些原则指导着这一重要的软件设计行为。

第二部分 阅读与翻译

A. 软件设计

设计是在[IEEE 610.12-90]标准中被定义为“定义一个系统或组件的体系结构、组件、接口及其他特征的过程”,也是“这一过程的结果”。作为一个过程来看,软件设计是一项软件工程生命周期活动。其中,软件需求被分析以产生软件内部结构的描述,将作为软件构建的基础。更确切地讲,软件设计(的结果)必须描述软件体系结构——即软件如何被分解和组织成组件,以及这些组件之间的接口。软件设计也必须在使其能够构建的详细级别上描述组件。

软件设计在软件开发中发挥着重要作用:它允许软件工程师创建出各种模型,这些模型形成了一种可实现的解决方案蓝图。我们可以分析和评估这些模型,以决定它们是否能够满足各项需求。我们也可以调查和评估各种候选的解决方案并进行权衡。最终,除了把它们作为构建和测试的输入和起始点,我们可以使用所产生的模型来计划随后的开发活动。

在软件生命周期过程的一份标准清单上,如 IEEE/EIA 12207 软件生命周期过程[IEEE 12207.0-96],软件设计包括适合于软件需求分析和软件构建之间的两个活动:

- **软件体系结构设计**(有时也称为顶层设计):描述软件的高层结构和组织,并确定各组件。
- **软件详细设计**:充分地描述每个组件以允许其构建。

一般策略

一些在设计过程中经常被引用的有用的一般策略的例子包括分而治之和逐步求精、自上而下和自底向上战略、数据抽象和信息隐藏、启发法的使用、模式和模式语言的使用、迭代和增量方法的使用。

面向功能(结构化)设计

这是软件设计的经典方法之一,在该方法中,分解关注于确定主要的软件功能,然后通过自上而下的方式对其进行详细阐述并完善。结构化设计通常用于结构化分析之后,从

而产生数据流图和相关的过程描述。研究人员已建议使用各种策略（如变换分析、事物分析）和启发式（如扇入和扇出、效力范围与控制范围）将数据流图转换为通常以结构图形式表示的软件体系结构。

面向对象设计

基于对象的许多软件设计方法已经被提出。该领域已经从 20 世纪 80 年代中期的早期基于对象的设计（名词等同于对象，动词等同于方法，形容词等同于属性），经过面向对象设计（在面向对象设计中，继承和多态扮演着重要的角色），发展到基于组件设计的领域（在基于组件的设计中，元信息可以被定义和访问（如通过反射的方式））。虽然面向对象设计的起源是数据抽象的概念，但责任驱动的设计也作为面向对象设计的替代方法而被提出。

以数据结构为中心的设计

以数据结构为中心的设计（如 Jackson 方法、Warnier-Orr 方法）从程序所操作的数据结构着手，而不是从其实实现的功能着手。软件工程师首先描述输入和输出的数据结构（如使用 Jackson 结构图），然后再基于这些数据结构图开发程序的控制结构。各种启发式方法已经被提出来处理特殊情况，如出现输入和输出结构不匹配的情况。

基于组件的设计（CBD）

软件组件是独立的单元，具有良好定义的接口以及可独立组合和部署的附属物。基于组件的设计处理与提供、开发及整合这些组件以改进复用相关的问题。

其他方法

其他有趣但非主流的方法也存在，如形式化和精确方法，以及转换方法。

B. 社交网络

社交网络指的是将一群个体组成特定的团体，就像是小型农村社区或地方分区那样。尽管社交网络可以通过亲身参与的方式来实现，特别是在公司、大学或中学这种场合，但它在网络上更加常见。

这是因为，与大部分中学、大学或工作场所不同，互联网上有着数以百万计的个人用户。他们在不断地尝试和别人建立联系，以获取或分享关于烹饪、打高尔夫、园艺、发展友谊、职业联盟、求职、企业对企业营销，甚至在互联网上，还有有关为“发展运动”烘制小饼干分享信息的群的第一手信息和经验。这些话题和兴趣点就像宇宙的故事一样丰富而多彩。

提到在线社交网络，网站是最为常见的方式，而社交网站也已经被大家所熟知。社交网络的功能就像是互联网用户的在线社区一样运作。通过网站的形式，这种在线团体的许多成员可以分享共同的兴趣爱好、宗教信仰、政治观点以及另类的生活方式。一旦加入了社交网络，就可以开展社交活动。这种社交活动包括阅读其他会员的个人信息页面或甚至直接联系他们。

交友只是在线社交网络的诸多益处之一。社交网络的另一个好处就是多样性，因为互联网允许世界各地的人们都可以接入社交站点。这意味着尽管你在美国，你同样可以与丹麦或印度的居民建立在线友谊。在这种情况下，你收获的将不仅是结交新的朋友，还可以学到新的文化以及新的语言。学习新的知识总是值得推崇的。

正如上文所提到的，社交网络总是能将特定的个人或团体组织到一起。尽管有些社交

网站有着特定的兴趣方向,其他的则不是这样。那些没有特定兴趣方向的网站,通常就是所谓的“传统”社交网站并且一般都开放了会员制度。这意味着任何人都可以成为这个网站的一员,而无关他们的爱好、信仰以及观点。然而,一旦你处于这个在线团体中,就开始经营自己的朋友圈,并且排除那些与兴趣爱好、奋斗目标不同的成员。

正如我们所熟知的,社交网络也可能会带来诸如数据盗窃和病毒感染等风险,并且这些风险正在上升。最为普遍的风险往往和互联网上的“掠夺者”或虚假身份有关。这些风险确实存在于在线网络上,但它们同时存在于真实世界。就像在俱乐部、酒吧、学校或单位遇到陌生人时一样,同样需要小心处理在线网络上的情况。

通过了解自己的互联网环境和交谈对象,可以安全的享受社交网络。可能需要打很多电话才能大致了解一个人,但是在见面之前永远都无法给对方一个清晰的评价。那么遵循常识,听从内心的感觉吧,会感觉到在线交谈中出现的异样。

当对于这个发现足够了解和感觉舒服,就可以开始从数百个社交网络当中选择加入。通过普通的互联网搜索方式就可以简单做到。这样的搜索可能会带来多个结果,包括 MySpace、FriendWise、FriendFinder、Yahoo! 360、Facebook、Orkut 以及 Classmates。

第三部分 模拟写作

软件设计规格说明书

设计过程的一个重要产品是一组描述将要被建成的系统的文档。正如人们所看到的,其中一部分要用自然语言告诉顾客和用户系统的功能;另一部分使用技术术语来描述系统的结构、数据和功能。因此,这两部分的内容可能会重叠,但是描述它们的方式也许不会重叠。

设计文档应包含一个章节称为“设计理念”的内容,概述在产生设计的过程中所考虑的关键问题和协调方式。这一指导观点有助于客户和其他开发者理解设计的各个部分如何以及为何会组合在一起。

设计还包括对系统组件的描述。有一个章节应该讨论用户是如何与系统进行交互的。

通常,一组图表或规范的符号描述系统的总体组织和结构,包括所有的抽象层次。

如果系统是分布式的,设计中的结构配置则要足够详细,以显示网络的拓扑结构,网络节点间如何相互访问以及各节点的功能分配。如果系统需求包含了时间方面的约束,或如果网络的节点必须同步,设计就要描述时钟是如何工作的。类似地,该设计还应包含有关消息控制和路由的信息;还可能会包括关于网络完整性的规定:确保数据准确,或是在失效后可以被恢复。

如果客户要求,设计的要素可能要讨论监测系统性能。此外,该系统可能会被手动操作,设计则要描述系统将如何工作。设计文档的其他章节可能会讨论故障的定位和隔离,系统重新配置,或是专门的安全措施。

最后,设计为需求提供前后参照,以证明其如何源自需求。这种一致性迫使我们去检查完整性和一致性。此外,这种前后参照将使得所做出的改进或修改之后更容易追踪。例如,如果需求变更,前后参照将指出所需的相应设计变更。

第六单元 系统实现

第一部分 听 与 说

A. 对话：创建高质量的代码

Kevin: 在编码之前，我认为有一点是必要的，那就是在编码过程中我们应该遵循一些基本原则。在这些指导原则的帮助下，我们将更可靠地编写高质量的代码并充分利用我们已决定用来编程的 C# 语言优势，C# 语言是一种流行的面向对象语言。

Jason: 我认为，对于面向对象的编程，程序员应该从类的角度来考虑程序设计。一个类就是具有内聚、明确职责的数据和程序的一个集合，即使不涉及公共数据，也可能是提供一组内聚服务程序的一个集合。要成为一名高效程序员的关键是，当影响任何一部分代码时，使可放心忽略的程序其余部分达到最大。类是完成这一目标的主要工具。

Sharon: 我完全同意。此外，我认为要编写高质量的类，首先并且可能也是最重要的一步是创建好的接口。这在于为接口创建良好的抽象，以表达和保证细节始终隐藏在抽象之后。特别是在系统分析和设计阶段，我们已经确定了所有的类，并为每个类定义了属性和相关的操作。现在，我们应该根据这些标准来评审并修改这些类，并且在编写所有类时都应该这样做。

Kevin: 是的。此外，另一个重要因素是类内部的高质量的程序。与任何编程语言的任何其他特性相比，这些程序使得整个程序更易阅读和理解。我认为，这也是有史以来为节省空间和提高性能所发明的最伟大的技术。如果我们每调用一次程序就要重复编码而不是将程序形成分支，很难想象我们的代码将会有多庞大。如果同样一段代码在多个地方出现而不是将其放在一个程序中，很难想象如果要对这段代码进行性能改进将会多么困难。一旦我们在第一步确定了类的主要程序，就必须构建每一个特定的程序。通常情况下，每个程序的构建都会引发对额外程序的需要，无论是次要的还是主要的，由编写那些额外程序所引起的问题通常会传回到类的整体设计。

Jason: 所以说编写类和编写程序更趋向是一个迭代过程。

Kevin: 没错。在每次迭代过程中，每当程序编写出来，我们都将对其进行测试。在类作为一个整体可操作之后，我们将类作为一个整体进行评审和测试，以找出那些在独立程序级别上测试不出来的问题。每一步，我们都应该检查做过的工作，并也鼓励其他人也进行检查。这样，我们将能够以最小的工作量投入、用最少的花费捕获错误。

Sharon: 另外，对于详细设计，伪代码程序设计过程是一个有用的工具，并能使编码变得更加容易。伪代码还能够被直接翻译成注释以确保注释准确有用。

B. 听力理解：编码

到目前为止，我们所学到的技能已能了解客户及用户所面临的问题，并为客户及用户

提供一个高级别的解决方案。现在,我们必须把关注点放在如何把解决方案实现为软件。也就是说,我们必须写出能够实现设计的程序。而由于以下几个原因,这项工作可能会困难重重。

第一,设计者可能对平台及开发环境的种种特点缺乏了解;用图表很容易描述的结构和关系不一定总是可以直接写成代码。

第二,必须以一种易于理解的方式来编码,不单单为了方便在测试的时候进行重读,也为了方便系统进化时其他人的工作。在现实世界中,大多数软件都是由团队开发,为了开发高质量的产品,需要进行各种不同的工作。即便是编写代码这一项工作,也通常也有很多人参与,并需要大量的合作与协调。因此,对于其他人来说,不仅能够理解所写的代码,还要理解为什么这样写以及这些代码如何与他们的工作相配合,这是非常重要的。要想成为一个真正出色的程序员,必须明白:编程首先是与其他程序员沟通,其次才是与计算机沟通。

第三,必须利用设计组织、数据结构以及编程语言语句的特性,来编写容易复用的代码。最重要的标准是,需要程序设计组件与程序代码组件之间的直接对应。如果设计的模块化没有被带入代码,那么整个设计过程便没有了任何价值。诸如低耦合、高内聚以及良好定义的接口等设计特性,同样也应成为编程特性,以便可以容易地在设计及编码间追踪和回溯算法、函数、接口以及数据结构。如果需求发生了变更,应首先反映在高层设计,然后通过低层设计一直追溯到必须修改的代码。

虽然大部分编码都是由个人努力所完成,但所有的编码在进行时都应考虑到整个团队。使用信息隐藏可以只显示组件的基本信息,以便其他同事可以调用或重用它们。应用标准可以加强团队成员间的交流。使用通用设计技术和策略可使代码更加容易测试、维护及重用。不管使用何种程序语言以及如何实现代码,都应该在编程时牢记上述这些软件工程常识。

C. 听写: 关注于重要的少数而不是不重要的多数

Pareto 原则(又被称为 80-20 原则)认为,对于很多事情来说,80%的结果源于 20%的原因。换句话说,可以用 20%的努力获得 80%的结果。这是一种统计方法来识别发挥最大作用的少数原因。

商业管理思想家 Joseph M. Juran, 质量管理领域的 Pareto 原则之父,推荐此原则,并以意大利经济学家 Vilfredo Pareto 的名字来命名该原则。Vilfredo Pareto 发现,意大利 80% 的收入集中到了 20%的人手中。这是一条可应用于多个领域的通用的经验法则。例如,在商业领域,被解释为“80%的销售额源于 20%的客户”。

类似地,有许多软件现象也遵循 Pareto 法则分布。例如,20%的模块占用了 80%的资源,20%的模块导致了 80%的错误,20%的错误消耗了 80%的修复成本,20%的模块占用了 80%的执行时间,20%的工具占到了 80%的利用率等。

这一原则在上述软件现象的明确应用之一是程序优化。Donald Knuth 用曲线图描绘了他的计算行数的程序,并发现有一半的执行时间花在两个循环上。他修改了几行编码,在不到一个小时内使曲线图的速度提高了一倍。这一事例表明,应该度量编码,找到关键点,然后将资源放在优化这些被使用最多的少数部分上。

Pareto 原则可以而且应该被每个聪明的人应用到日常生活中。这一原则可以使我们的收益和效力翻倍。

第二部分 阅读与翻译

A. 计算机程序设计

程序就是一个计算机必须遵循的将数据转化为信息的指令列表。指令是由编程语言中的语句组成,如 Basic 语言。程序的例子包括文字处理、桌面排版或薪金处理。程序设计是用于创建指令列表的由如下 5 个步骤组成的过程。

1. 问题阐明(定义)步骤由 6 个小步骤组成——明确程序目标和用户、输出、输入和处理任务;研究程序的可行性;以及撰写分析文档。

2. 要设计解决方案首先需要建立一个算法。算法是一种解决某一特定问题的公式或步骤的集合。在程序设计这一步,软件设计被分为两个小步骤。首先,该程序的逻辑通过自顶向下方法和模块化确定,使用层次图(或结构图);然后是详细或叙述形式的设计,使用伪代码或图形化地使用流程图。伪代码是设计使用标准人类语言表述来描述逻辑和处理流程的设计程序的一种方法。大多数程序员使用称为结构化程序设计的设计方法。结构化程序设计采用将程序分解成模块形式的自顶向下方法。在结构化程序设计中,顺序、选择和迭代(或循环)三种控制结构用来形成程序的逻辑。

3. 一旦设计完成,实际的程序编写也就开始了。编写程序即是所谓的编码。编码被很多人认为就是程序设计,其实这只是其中的 5 个步骤之一。编码在于将逻辑需求由伪代码或流程图翻译成编程语言——组成该程序的字母、数字和符号。一种编程语言是一组命令计算机操作的规则。众所周知的编程语言的例子有 C、C++ 和 Java。这些都是高级语言。

4. 程序测试包括运行各种测试,然后运行真实世界的的数据,以确保程序能正常工作。两个主要活动是桌面检查和调试。这些步骤被称为 α 测试。桌面检查只是简单走查或检查程序,来确保它没有语法和逻辑错误。一旦该程序已经完成桌面检查,进一步的错误或 bug(臭虫)肯定会显现出来。调试就是检测、找到并删除程序中的所有错误。错误可能是语法错误或逻辑错误。语法错误是由印刷错误和不正确使用编程语言造成的。逻辑错误是由不正确使用控制结构造成的。桌面检查和调试之后,该程序也许就可以在实验室里运行得很好了。不过,它还需要用真实数据进行测试,这就是所谓的 β 测试。

5. 撰写程序文档是程序设计中的第 5 步。结果文件由程序是什么以及如何使用程序的书面描述组成。文档不仅是程序设计的最后步骤,而且已经(或应该已经)贯穿程序设计的所有步骤。文档由用户文档、操作员文档和程序员文档组成,可以用来维护程序。

编程语言已经历 5 代,分别如下:

第一代,1945 年——机器语言。

第二代,20 世纪 50 年代中期——汇编语言。

第三代,20 世纪 50 年代中期到 20 世纪 60 年代初期——高级语言(过程语言和面向对象语言),如 Fortran、COBOL、BASIC、C 和 C++。

第四代,20 世纪 70 年代初期——更高级的语言(面向问题的语言),如 SQL、Intellect、NOMAD、FOCUS。

第五代, 20 世纪 80 年代初期——自然语言。

编译器是一种语言翻译程序, 可在计算机执行程序之前将整个高级语言程序转换成机器语言。应用编译器的过程语言的例子有 COBOL、FORTRAN、Pascal 和 C。解释器是一种能将每种过程语言语句转换成机器语言并立即执行的语言翻译程序。BASIC 是一种使用解释器的程序语言。当使用编译器时, 在程序执行之前需要两个步骤(源代码和目标代码), 而解释器只需要一个步骤。编译器程序语言的优势是, 一旦获得目标代码, 程序将执行得更快; 另一方面, 解释器程序语言的优势是使程序更容易开发。一些语言翻译器——如 C++ 和 Java 所使用的——既可以编译又可以解释。

在面向对象程序设计中, 数据和用于处理数据的指令组合成为一个自给自足的可用在其他程序中的“对象”。一个对象就是一个自我包含的模块, 该模块是由预装的编程代码构成。当一个对象的数据要被处理时发送“消息”。消息是当包含对象的一个操作需要被执行时发送给该对象的一条警告。消息只需要识别操作。实际如何执行被嵌入到作为对象一部分的处理指令中。这些对象中的处理指令被称为方法。

面向对象程序设计有三个重要的概念, 分别是封装、继承和多态。封装是指对象包含(封装)了数据和相关的处理指令。一旦对象已经创建, 它就可以在其他程序中复用。对象的使用还可以通过类和继承的概念来扩展。所有相互派生或关联的对象形成一个类。每个类包含了本组独一无二的特定指令(方法)。继承是一种将对象的特性从层次中的类到子类向下传递的方法。因此, 新的对象可以通过从现有类中继承特性来创建。多态性是指一条消息(广义的请求)基于它要被发送的对象产生不同的结果。

面向对象编程语言的例子有 C++ 和 Java。C++, 加号代表“比 C 语言更好”, 在传统的 C 语言中融合了面向对象的功能。C++ 是由 Bjarne Stroustrup 设计的。通过 C++, 程序员可以不用面向对象的特点撰写标准的 C 语言代码, 也可以使用面向对象的特性, 或使用两者混合的代码。Java 高级编程语言由 Sun Microsystems 公司在 1995 年推出, 用来编写能够从因特网下载并立即在多种计算机上执行的简洁的程序。Java 与 C++ 类似, 但被简化了, 以消除引起常见编程错误的语言特性。

标记语言是一种编码或“标签”, 被插入到嵌入了有关文本外观和结构细节的文本中。HTML(超文本标记语言)是一种使人可以为因特网创建屏幕显示文档的标记语言, 这些文档可以轻松地通过文字和图片链接到其他文档。VRML(虚拟现实建模或标记语言)是一种用来创建三维的网页的程序语言, 包括交互式动画。XML(可扩展标记语言)是一种使用 SGML 编写的元语言(用来定义其他语言的语言), 可以使在因特网上的文档更容易地进行互交换。

脚本是一种被嵌入到完成特定函数或程序的网页中的自己执行命令的短列表。JavaScript 是一种被 Web 浏览器广泛支持的流行的面向对象脚本语言, 使得 HTML 网页增加了交互功能, 否则 HTML 网页是静态的。ActiveX 是一套控件或可复用组件, 可以使程序或几乎任何形式的内容嵌入到网页中。Perl(实用报表提取语言)是一种为文本操作而开发的通用编程语言, 现在也用于 Web 开发、网络编程、系统管理、GUI 开发以及其他任务。

B. 基于位置的服务

基于位置的软件可以获取当前位置并由此分析出最近的商店、餐厅、剧院以及其他地

点的信息。想要找到最近的 ATM? 需要找到前往机场的最佳路线? 想要和附近的朋友聊天? 想要和某些用户分享发现的墨西哥餐厅? 现在的基于位置的软件可以回答这些问题并可以提供更多服务。

基于位置的软件可以在台式机、笔记本电脑以及掌上电脑上使用。移动设备可以通过内嵌的 GPS (全球定位系统) 或分析测量与地面基站之间的距离来精确地找到位置。台式机和笔记本电脑可以根据网络供应商和附近的专用计算机网络来确定位置。

当设备记录位置后, 这些数据也有可能被滥用。需要谨慎对待那些可以跟踪所在位置的设备和软件。在一些情况下, 可以暂时或永久的关闭它们; 而在其他情况下, 这些追踪定位设备可能不在控制之下, 而必须在设备和软件所能提供的服务与自己的隐私之间进行取舍。

基于位置的服务 (LBS) 是一种计算机程序级别的服务, 用来将时间和位置数据的特定控制作为控制特性包含在计算机程序中。这种基于位置的服务是一种信息服务并且在当前的社交网络中以娱乐服务的形式拥有一定数量的用户。这些用户可以通过移动设备的移动网络或地理位置信息来使用这种服务。同样, 随着智能手机和平板电脑市场的扩张, 这种服务变得越来越重要。

LBS 的应用范围很广, 如健康保障、室内物品搜索、娱乐、工作、个人生活等。

LBS 包含用于确定人或物所处位置的服务, 如找到最近自动取款机、找到朋友或员工的所在之处; 包含包裹追踪以及车辆追踪的服务; LBS 还包括移动商务, 如基于用户的当前位置来发布广告和优惠券; 还包括个性化天气服务以及基于位置的游戏。它是电信一体化的一个实例。

基于位置的服务, 如 Foursquare、Gowalla、Where 以及 Facebook Places 使用移动电话或智能机的地理位置定位功能来向用户提供信息和娱乐。

这些服务允许客户在任何地方“签到”, 如咖啡厅、盛典现场、餐厅、旅店、机场甚至私人住所, 从而换取真实或虚拟的奖励。

在一开始看来, 认为人们会“签到”的想法或许似乎很疯狂, 但他们确实这么做了 (而且人数非常多)。事实上, 人们会“签到”源自多方面原因, 包括告诉他们的朋友自己的位置、记录自己的行踪、赢取诸如徽章和市长职位 (或头衔) 的游戏奖励, 以及从参加奖励“签到”的商业活动和场所中来解锁特价商品。

作为一个商人, 可以利用这些特点来提供特价商品、打折商品以及对忠诚客户的奖励。

LBS 真是太奇特了。

第三部分 模拟写作

进度报告

进度报告的作用和内容

进度报告在工程学中应用很广。一旦已经有了一个成功的提案并且已经获得相应的资源来开发某个项目, 那么就需要向客户及时汇报项目的进展情况。这些进展情况通常用进度报告来汇报。进度报告可以是一页的备忘录或是多页的信件、简短报告、正式报告及演示。这样的报告要针对分配项目的人。报告的目标是对已完成的项目部分及未来的工作计

划进行公正的评价,并使项目经理或项目资助人能据此决定该项目的未来。然而,无论项目的规模或是重要性如何,都会碰到一些阻碍:额外的需求、错误传达的信息、难题、延期或意料之外的支出。进度报告必须说明这些阻碍,应该讨论在项目中正在遇到的困难以及所需的进一步的帮助。

进度报告中应包含的内容取决于具体的情况,但是大多数报告都有以下类似内容:

1. 项目的背景。
2. 自从上次报告以来所取得的成果。
3. 新出现的问题。
4. 未来要做的工作。
5. 评估自己是否将按照预订的进度和预算完成项目。

进度报告的组织形式

进度报告的结构应当取决于项目的原始提议:利用原有的里程碑或时间安排。了解这一点之后,一个更加全面的成分列表可以提供更加清晰的结构,具体如下:

1. 简介

通常情况下,应当说明这份报告的目的以及目标受众,应当清楚地定义报告所覆盖的时间范围,然后解释该项目、项目目标以及自上次报告以来的项目进展状态。

2. 项目描述

在简短的报告中,这部分内容可以包含到简介中。但是如果将其列为一个独立标题,那么熟悉该项目的读者就可以跳过这部分内容;而对项目不甚了解的读者,需要从中了解一些概要性的细节问题,如项目的目的和范围、起止时间以及参与的各方等。

3. 已完成的工作

这一部分应遵循该项目提案进度中所提出任务的进展,这是报告的主体。这部分可以采用项目-任务法、时间法或是综合法来进行陈述。

- 项目-任务法:这种方法关注的是任务。已定义的里程碑可以将报告在逻辑上组织成这种结构。如果同时进行多个半独立的任务,同样可以使用这种方法。
- 时间法:这种方法关注的是时间。如果时间表(或最后期限)比里程碑更重要,则使用这种方法。同样,此方法也适用于简单的线性结构的项目。
- 综合法:上述两种方法也可以结合起来使用。例如,在先前的工作中,可将已经完成的部分分成单个任务。即在任务中,关注哪些部分已经完成,哪些部分还在进行中,哪些部分尚未进行。

具体的项目将决定选用哪种方法。如果项目中遇到的困难或是所需的变更与时间有关,那么使用时间法会更有利;类似地,如果遇到的问题或是变更与特定的任务相关,那么就使用项目-任务法。此外,该部分还可以附上财务数据概要,可以使用表格或是附录的形式,或作为一个独立的章节。

4. 遇到的问题

进度报告没有必要仅为了给客户看。正如开始所提到的,问题是在报告中希望提到的。不必将这些问题隐藏起来;应该解释清楚碰到了什么问题以及这些问题对项目的关键部分(如时间安排、价格或是质量)有何影响。通常情况下,工程师或专家会从此报告中得到帮助,因为我们与客户共享或提醒了客户项目中出现的问题。所以,如果这个问题是过去出

现过的，那么可以说明过去是如何克服的。这个是一个很小的问题；实际上，却让我们看起来很出色。如果这个问题尚未解决（就在眼前或是将会出现）并且有能力解决，那么应解释将希望如何克服它。

5. 需求的变更

在这一部分，要记录项目中的变更：增加的里程碑、新的需求或是进度的变更（好的或坏的）。即使这些变化不会影响到项目的最终目标，仍需要告诉项目的赞助人这些问题是如何解决的。

注意：如果这些变更是项目中遇到的问题直接引起的，第4和第5部分可以合并。这样需要对结构进行一些调整，即第一个问题及它所需要的变更，然后是第二个问题及其需要的变更，以此类推。

6. 工作进度

在这一部分，要讨论为了完成项目目标所制订的计划。在多数情况下，进度报告的这一部分写成与提案中“行动规划”部分相同的方式。区别在于，现在应该比那时对进度和成本有了更好的认识。这部分，像“已完成的工作”部分一样，根据项目实际情况，可以使用上述提到的三种方法（项目-任务法、时间法或综合法）中的一个。

7. 项目总体评估

由于进度报告并不是针对一个已完成的项目，因此结论部分仅需要对该项目目前的进展情况进行专业评价。不现实的乐观或是过度悲观都是不恰当的。不要随便承诺项目将提早完成，因为一个挫折可能会耗费大量时间。同样地，如果落后于进度，也不要过度担心，因为在以后的进程中还可以争取到时间。对工程师来说，更重要的是说明一切可能影响最终产品的期望质量的各种因素。请牢记，进度报告的目的是，使项目经理或赞助人能够据此做出决策。

第七单元 系统测试

第一部分 听 与 说

A. 对话：软件测试

Kevin: 似乎我们还没有花足够的时间来讨论测试呢。

Sharon: 没错，但我们一直都在考虑测试。事实上，不止在考虑。我们在需求阶段计划的测试策略采取了一种增量方法。这种方法从个体程序单元的测试开始，逐步到为了便于单元集成所设计的测试，最后到检查所构建的整个系统的测试。

Kevin: 是的，在编码开始之前，我们已经完成了针对系统每个组件的单元测试的设计，这是敏捷编程的一种首选方法。即使我们没有采用极限编程，这仍然是一个很好的想法——对设计信息的评审为建立测试用例提供了指导，这些测试用例可能会显示出之前所讨论的每一类错误。现在，我们可以使用这些测试用例来进行单元测试了。

Jason: 在集成测试阶段，可以使用一些经典的集成测试策略，如自上而下法或自底向上法。

除了那些用于传统的、层次结构软件的方法，也可以使用体系结构驱动的方法：根据需求规格说明书，基于已确定的功能部分，集成软件组件或子系统。

Kevin: 说得好。在集成测试期间，可以同时制定一个验证策略。在这一步中，应该根据软件需求规格说明书定义验证所需的测试及其标准，包含在其验证标准部分中的信息形成了验证测试方法的基础。

Sharon: 可能我们还得进行一些高级别的测试，如恢复测试、安全测试、压力测试和性能测试。最后，我们将同客户一起协调验收测试工作。

B. 听力理解：软件测试

测试在软件质量保证中扮演着重要的角色。由于前期验证手段的限制，设计和需求的错误仍会在代码中出现。测试用于检测这些错误以及编码阶段引入的错误。

测试是一种用于验证及确认的动态手段，在测试中，运行被测试系统且观测系统行为。因此，测试可以察觉到系统失败，以此推断错误的出现。然而，必须执行单独的活动来识别错误并排除它们。

有两种测试手段：黑盒测试及白盒测试。在黑盒测试中，不考虑被测试系统的内部逻辑，根据规格说明或需求确定测试用例。黑盒测试通常被称为功能性测试。等价类划分、边界值分析以及因果图例是用于选择黑盒测试用例方法的例子。基于状态的测试是另一种测试方法，在这种方法中，系统建模为一个状态机，然后这个状态机模型用于使用某些转换或基于路径覆盖的标准来选择测试用例。基于状态的测试也可被视为灰盒测试，因为这种测试通常需要除需求之外更多的信息。

在白盒测试中，测试用例的选择完全取决于被测试程序或模块的内部逻辑，而不考虑外部规格说明。标准通常是明确指定的，但选择测试用例的步骤是留给测试者的。变异测试是白盒测试的一种方法，这种方法通过改变原始程序创建原始程序的突变体。测试标准是通过让变种从原始程序生成不同的输出来消灭所有的突变体。

由于测试的目的是检测程序中的任何错误，通常会使用不同级别的测试。单元测试用于测试一个模块或一个小的模块集合，主要关注于检测模块中的编码错误。在集成测试期间，模块被组合成子系统，并对子系统进行测试。集成测试的目的是测试系统设计。在系统测试和验收测试中，对整个系统进行测试。系统测试和验收测试的目的是对照需求测试系统，以及测试需求本身。白盒测试可被用于单元测试，而更高级别的测试往往使用黑盒测试。

C. 听写：冒烟测试

在电子学中，电路首次通电，如果设计或配线发生错误则有可能出现真的烟雾，这被称为“冒烟测试”。冒烟测试是指在维修或是首次组装后进行的首次测试，以保证测试的系统不会出现致命性的错误。

在软件中，术语“冒烟测试”喻指描述在变更被加入产品的源代码树之前确认代码变更的过程。在代码评审之后，冒烟测试是用于识别和修复软件中缺陷的最具成本效益的方法。它被设计以证明代码功能中的变更以期望的方式运行，且不会破坏整个编好的程序的稳定。

《代码大全》一书的作者 McConnell 说：“无论你选择哪种整合策略，集成软件的一种好方法就是‘每日编好的程序和冒烟测试’。”每天，每个文件都经过编译、链接并组合形成一个可执行程序作为一个“编好的程序”；然后程序完成“冒烟测试”，这是一种对整个系统从头到尾的相对简单的检查，不必详尽无遗，但要能够暴露阻碍编好的程序正确运行功能的主要问题。

这个简单的过程当用在复杂的、时间严格的软件工程项目时会提供相当多的好处。首先，通过每天管理代码并及早发现质量问题，可以将集成风险降到最小。它也简化了缺陷诊断和修正，因为新加入到编好的程序中的软件程序很可能就是新发现错误的原因。此外，它还鼓励小小的进展评估，当开发人员通过每日编好的程序每天看到产品有更多的进展时，也可以增强士气。

第二部分 阅读与翻译

A. 软件测试

测试是一项用于通过识别缺陷和问题来评估和改进产品质量而进行的活动。软件测试的观点向着更具建设性的方向发展。测试不再被视为在编码阶段完成后才进行，并仅仅以检测故障为目的而开展的一项活动。软件测试现在被视为一项围绕整个开发及维护过程，并且其本身也是实际产品构建的一个重要组成部分的活动。确实，测试计划应该在需求过程的早期阶段开始，并且随着开发的进行，测试计划及过程必须系统化地不断制定并尽可能地完善。这些测试计划和设计活动本身为设计者在突出潜在缺陷方面（如设计疏忽或矛盾，以及文档中的遗漏和二义性）提供了有用的信息。

软件测试通常伴随着开发和维护过程以不同的级别执行。也就是说，测试的对象是不同的：一个单独模块，一组这样的模块（目的、用途、行为或结构相关的），或者整个系统。三大测试阶段可从概念上进行划分，即单元测试、集成测试和系统测试，三个阶段同等重要，不包含过程模型。

单元测试

单元测试用来在隔离的可分开测试的软件组件中验证功能。根据上下文的不同，这些软件组件可以是单个子程序或更大的由紧密相关单元组成的组件。通常，在调试工具的帮助下进入被测试代码时即发生了单元测试，并且有可能有写代码的程序员参与。

集成测试

集成测试检验软件组件之间交互的过程。典型的集成测试策略，如自顶向下法或自底向上法等被用来测试传统的、层级化结构的软件。现代系统化的集成策略更倾向于体系结构驱动的，指基于已识别的功能性部分来集成软件组件或子系统。

系统测试

系统测试关注整个系统的行为，且通常被认为适用于将系统与非功能化的系统需求（如安全、速度、准确度和可靠性）进行比较。与其他应用程序的外部接口、实用程序、硬件设备或操作环境也会在这个层面进行评估。

测试的其他重要目标还包括（但不限于）可用性评估和验收，对此会采取不同方法。注意，测试目标会随着测试对象的不同而变化，一般而言，不同的测试级别有不同的测试

目的。

验收/合格测试

验收测试对照客户需求检查系统行为。尽管这些可能已经说明过了，客户还是需要进行或指定一些典型任务来核对其需求已经得到满足或者组织已经为软件的目标市场确定了这些。这一测试活动可能也可能不涉及系统开发人员。

α 测试和 β 测试

在发布之前，软件有时先提供给潜在用户的少量代表试用，这可以是内部的（ α 测试）也可以是外部的（ β 测试）。这些用户会报告产品的问题。 α 及 β 的使用通常是不受控的，且不是总在测试计划中提到。

很多技术已经被开发用于执行测试，通过运行一个或多个从已识别的、被认为是等价类的执行中获得的测试，尝试“破坏”程序。作为这些技术基础的首要原则是，在识别一系列有代表性的程序行为时尽可能地系统化。例如，要考虑输入域的子类、场景、状态和数据流。很难找到一个划分所有测试技术的一致基准，这里使用的方法可以被视为一种折中。这个分类基于测试是如何从软件工程师的习惯和经验、规格说明、代码结构、待发现的（真实的或人工的）错误、领域用途或最终的应用本质中得来的。如果测试依赖于关于软件是如何被设计或编码的信息，这些技术则有时被归类为白盒测试或玻璃盒测试；如果测试用例依赖于输入输出行为，这些技术则被归类为黑盒技术。最后一个类别涉及以上两种或更多技术的组合使用。显然，这些技术通常不能被所有从业者均等使用。

B. 通用计算图形处理器

GPGPU（通用计算图形处理器）通过使用图像处理单元处理来实现高性能计算。与其他高性能计算（HPC）算法相比，使用高性能专用图像处理器图形算法的特点显而易见。同样的专用硬件设备，也可以对这些算法进行加速。

适用于 GPGPU 的算法具有两种特点：数据并行以及吞吐量密集。数据并行是指处理器可以同时处理不同的数据元素。吞吐量密集是指算法可以处理大量的数据元素，以至于并行处理可以广泛实现。通过利用这两个特点的优势，GPU 可以通过将数百个相互关联的小处理单元组合到一起来实现多数据元素同步处理，从而达到最佳性能。

不难想象，基于像素的应用（如计算机视觉、视频、图像处理）非常适合于 GPGPU 技术来处理，因为这个原因，很多这些领域的商业软件包都包含了 GPGPU 加速。经常依靠偏微分方程（PDE）的数字解决方案以及依靠大量使用线性代数运算的物理仿真应用，同样也非常适合使用 GPGPU 技术。在渲染领域，光线追踪（一种用于高真实度图像合成的计算密集型算法）使用 GPGPU 技术来计算光线物体交叉，从而实现实时光线追踪。

GPU 的特性使得其处理器结构与传统 CPU 明显不同。CPU 使用了大量资源（主要在芯片区）使得单一指令流高速运行，并使用了高速缓存来降低内存延迟以及复杂的指令流处理方法（乱序执行、推测执行的流水线操作）。与此不同的是，GPU 使用芯片区处理数百个单独的处理元素，从而在多个数据元素中同时处理单一指令流。内存延迟会被非常快速的上下文转化所掩盖：当数据元素的一个子集在处理过程中需要进行存储器读取时，这

个子集会被搁置并让另一个无须在存储器参量中等待的子集运行。

GPU 可以在任何地方用 CPU 的 10 倍到 100 倍或更快的速度来运行特定算法,这是 GPU 的一个很大优势。

GPU 存在两点不足:

1) 为了达到这样的提速,算法必须根据 GPU 的结构来编码,而针对 GPU 的编程与针对传统的 CPU 有很大的不同。特别是对现有代码而言,让它们适用于 GPU 加速的难度远远大于在不同类型 CPU 之间的切换。一个有经验的 GPU 程序员需要对这些代码进行深入研究,并对关键部分进行大规模修改。

2) 将 GPU 硬件加入到系统中会增加功耗、发热以及费用。有时,增加可使用 CPU 数量的方法会更加经济实用。

GPGPU 计算在很多应用领域对高性能计算产生了巨大冲击。在天气预报、分子运动、流体动力学等领域广泛应用的 HPC 代码正在通过 GPGPU 加速技术来进行更新。常见的商业科学以及工程技术的相关应用已经提供 GPGPU 加速:特别是 MATLAB 和 ANSYS,以及诸如 AMBER、LAMMPS、NAMD 的开源系统,Gromacs 也开始使用 GPGPU 技术。更为重要的是,用于尖端科学和工程研究方面的代码开发已经广泛接受 GPGPU 技术。

高性能计算长期以来通过并行系统来满足尖端科学和工程研究方面的需求。随着 GPGPU 技术的实现,单个节点的能力大大增强,但仍然不能为当前的 HPC 应用提供足够的性能。这些应用目前仍依赖于数万个单独 CPU 同时工作的系统,来获取所需要的性能。

第三部分 模拟写作

软件测试规格说明书

测试计划描述了将测试整体分解成能处理特定项的单个测试。对于每个这样的单个测试,都要写一份测试规格说明书和评估。规格说明书应以需求列表开始,这些需求的满意度将由测试说明。参照需求文件,这一章节解释测试的目的。

使用表格或图表是一种查看需求与测试之间对应关系的方法。将测试所涉及的系统功能列举在表格中。性能测试也可以采取类似的方法,不过列举的不再是功能需求,而应是访问速度、数据库安全等有关的需求。

通常情况下,一个单个的测试实际上是一些更小测试的一个集合,这些更小测试的总和举例说明了需求满意度。在这种情况下,测试规格说明书应该展示这些更小测试与需求之间的关系。

每个测试都应由测试原理来指导,而且要采用一系列方法。但是,原理和方法可能会受到其他需求和现实测试环境的约束。测试规格说明书应该清晰说明这些测试环境。可能有如下情况:

- 系统正在使用的是来自用户或设备的实际输入,还是特殊情况是由程序或代理设备产生的?
- 测试覆盖标准是什么?
- 数据应该如何记录?

- 测试是否有时间、接口、设备、人员、数据库或是其他限制?
- 如果测试包含一系列较小的测试,那么这些小测试应当以何种顺序执行?

如果在评估前,需要处理测试数据,那么测试还要考虑到处理程序。例如,如果一个系统产生了大量的数据,那么有时需要使用数据压缩技术来处理这些输出,以使结果更适于评估。

伴随每一个测试是一种判定测试何时完成的方法。因此,测试规格说明书之后还要详述如何得知测试已经结束以及相关需求已经得到满足。例如,计划解释了什么范围的输出结果是满足需求的。

评估方法在完成标准之后。例如,可以将测试期间产生的数据收集到一起,进行人工比较,然后再由测试组进行检查。或者,测试组可以使用自动化工具来评估部分数据,然后检查总结报告或与预期输出结果进行逐项比较。

第八单元 系统交付

第一部分 听 与 说

A. 对话:软件部署

White 先生: 欢迎来到酒店!感谢你们为我们四季酒店管理信息系统开发所做出的努力。

Kevin: 不客气。今天,我们将向你们交付软件,包括所有经过测试的程序文件、数据文件 and 用户补充文档,如用户指南和操作手册。在我们的软件中还添加了超文本“帮助”文件和疑难问题排除指南。

White 先生: 听起来太棒了!那么我们可以在系统部署后马上运行了吗?

Kevin: 没问题。但我建议还是首先在一个小群体的用户代表中测试软件。在确保运行正常之后,可以在整个酒店中部署系统。通过这种方式,这些用户代表可以先运行该软件并在实际中找出问题,而问题的影响将控制在一个较小范围内并且问题可以及时地得到解决。这也是一种常见的软件部署方式。

White 先生: 真的吗?我觉得是个好主意。但多大的范围合适呢?我们总共有 100 名工作人员和 20 台计算机。

Sharon: 在大概三分之一的酒店信息台上部署就可以。我认为,信息台处理大部分酒店日常业务,并能有力高效地发现和揭示潜在的问题。

White 先生: 好的。我完全同意这个建议,并会尽快安排设备和人员。此外,如果发现问题时,恐怕我们还将需要一些及时支持。

Kevin: 不必担心。我们将提供安装和启动帮助,作为一个支持小组,我们将为您确保疑难问题排除帮助,直到系统正常运行半年。

Jason: 这是我们的联系列表, 上面有电话号码和电子邮件地址。另外, 我们也需要一份酒店负责该系统工作人员的联系信息, 以便及时联系。

White 先生: 没问题。我会在两天之内将我们的联系信息发电子邮件给你们。

Kevin: 关于这方面, 更进一步地, 我认为我们最好正式地建立问题记录和反馈机制, 包括“反馈过程是什么样的?”、“你们如何记录问题并将其报告给我们?”、“使用哪种形式, 纸介和/或电子的?”等。这样, 就可以收集和记录反馈意见, 与你们进行沟通和评估, 以一起确定修改方案。

White 先生: 好的, 那我们继续吧。

B. 听力理解: 软件交付

软件增量交付对于任何一个软件项目来说都是一个重要的里程碑。不过在为此高兴以前, 应注意以下重要的几点必须完成。首先, 应保证交付物已经过测试和修改并且没有 bug。其次, 交付物应不止是软件本身。也就是说, 除了可执行的软件以外, 支持数据文件、支持文档以及其他相关信息必须被集中起来一并提交。必须向最终用户提供适当的指导性材料, 如培训帮助、疑难问题排除指南和“该软件增量与前交付版本相比不同之处”的描述。并且, 在软件交付之前, 必须建立起一套支持体系制度。最终用户希望在问题发生时能得到反应迅速且准确的帮助信息。如果支持只是临时的, 甚至根本就没有, 客户将马上变得很不满意。应该对支持进行计划、准备好支持材料, 并建立适当的记录支持体制, 以使软件团队能够对所请求的支持类型进行明确的评估。

已交付软件为最终用户提供益处和帮助, 同时也为软件开发团队提供有用的反馈。当增量投入使用后, 应鼓励最终用户对软件的功能、特性以及易用性、可靠性和任何其他各种适当的特性做出评价。软件开发团队应收集和记录用户的反馈, 并用于对已交付增量做出及时的修改, 定义要合并到下一个已计划的增量中的变更, 针对这些变更相应地制定必要的修改设计, 并为下一个增量修订计划以反映这些变更。

C. 听写: Bug 与调试

自 14 世纪以来, bug 一词是用来形容“妖怪”的。在计算机领域, 人们普遍认为 bug 一词来自于由海军上将 Grace Hopper 博士。他是 COBOL 的发明人, 在 20 世纪 40 年代发现了第一个计算机 bug。尽管不像妖怪一样会飞, 但是软件中的缺陷可能会将自己呈现出各种各样的形式, 从误解需求到编码错误。

在进入计算机领域之前, “调试”一词是用于航空领域的术语, 在 1952 年美国计算机协会国家会议的三篇论文中首次使用。在评估测试结果时, 软件工程师经常会遇到软件的“症状”迹象。也就是说, 这个错误的外部表现和内部原因相互之间可能并没有明显的联系。能将症状与原因联系到一起的难以理解的思考过程就叫“调试”。具体来说, 在计算机领域中, 调试是指找出并修复计算机程序代码中或是硬件设备管理中的 bug 的过程。无论是商业产品还是企业或个人应用程序, 在几乎任何新的软件或是硬件开发过程中, 这都是一个非常必要的过程。

没有人能写出完美的软件, 因此这里假设调试将会占据写软件的大部分时间。在《调

试技术》一书中, David, J Agans 为找到最难捕获的软件和硬件问题提供了以下 9 条不可缺少的原则: 理解系统、让系统失效、停止思考开始观察、分而治之、每次只修改一个地方、记录审核踪迹、检查插座、重新审视。如果这样的问题都没有解决, 那么也就解决不了了。

第二部分 阅读与翻译

A. 软件维护

系统开发在系统可操作时, 即系统被最终用户在真实生产环境中使用时, 宣告完成。任何在系统投入使用后对系统所做的修改都被认为是维护。许多人将软件系统维护视为同硬件维护一样: 修复或防止坏掉的或不正确工作的零件。然而, 软件维护并不能以同样的方式看待, 下面看看为什么。

软件工程的目标之一就是开发能准确定义问题、将系统设计为解决方案、实现正确有效的程序集并测试系统错误的技术。这个目标与硬件开发者类似: 制造能够按照说明工作、可靠、无错误的产品。在这样的一个系统中, 硬件维护集中于替换坏掉的零件或是应用技术来延长系统的使用寿命。然而, while do 语句并不会在 10 000 次循环后坏掉, 分号也不会从语句末尾跑掉。与硬件不同, 软件不会退化, 也不需要定期维护。因此, 软件系统与硬件不同, 不能把对硬件系统的维护经验完全类推用在软件工程的其他方面。

软件系统与硬件系统之间的最大不同在于, 软件系统是为融入变化而开发的。除了最简单的情况, 开发的系统都是进化的。也就是说, 一个或多个系统所定义的特性通常会在系统生命周期中发生变化。

软件系统可能不仅仅因为客户做出决定改变做事方式而变化, 也会可能因为系统本身的性质发生变化而变化。例如, 考虑一个公司计算薪金削减和发放薪水的系统。该系统取决于该公司所在的城市、州或省及国家的税法或规定。如果税法变了, 或如果该公司搬到了其他地方, 该系统可能就需要修改了。因此, 即使系统过去一直可接受地工作, 现在的系统也可能需要变更。

在开发系统时, 将主要关注点放在正确实现需求并运行的编码上。在开发的各个阶段, 开发团队不断地参照前面阶段的工作。设计组件约束于需求规格说明, 代码组件被对照和评审以遵从于设计, 测试基于找出功能和约束是否依据需求和设计工作。因此, 开发需要仔细、有控地回溯。

维护就不同了。作为维护者, 不但要回头看开发的产品, 而且要看眼前的工作, 通过与用户和操作人员建立一种工作关系以发现他们是否满足于系统工作的方式。也要向前看, 预估可能会出现的错误, 考虑因变更的业务需要而要进行的功能变更, 也要考虑因变更的硬件、软件或接口而需要的系统变更。因此, 维护的涵盖的范围更广, 需要更多的追踪及控制。让我们来仔细看看为保持系统顺畅运行所需进行的活动, 并看看由谁来执行这些活动。

维护活动与开发活动相似: 分析需求、评估系统和程序设计、编写并评审代码、测试变更, 以及更新文档。因此执行维护的人——分析员、程序员及设计员——扮演着类似的

角色。然而，因为变更通常需要对代码结构和内容详细了解，因此程序员通常在维护中比在开发中扮演更重要的角色。

维护通常同时关注系统进化的如下 4 个主要方面：

- (1) 维护对系统日常功能的控制。
- (2) 维护对系统变更的控制。
- (3) 完善现有可接受的功能。
- (4) 防止系统性能下降到不可接受的层级下。

通常，彼此独立的分析员、程序员及设计员（有时包括一到两个开发团队成员）小组被选定为维护团队。一个崭新的团队可能比原来的开发者更客观。

维护系统涉及团队全体成员。通常，用户、操作员或客户代表带着评价或问题与维护团队接洽。分析员或程序员要确定代码中哪些部分受到了影响、其对于设计的影响以及可能的资源（包括时间和工作量），以做出必要的更改。团队要参与的许多活动如下：

- (1) 了解系统。
- (2) 在系统文档中查找并定位信息。
- (3) 持续更新系统文档。
- (4) 扩展现有功能以适应新或变化的需求。
- (5) 添加新的系统功能。
- (6) 找到系统失败或问题的来源。
- (7) 定位并改正错误。
- (8) 回答有关系统工作方式的问题。
- (9) 重构设计和代码组件。
- (10) 重写设计和代码组件。
- (11) 删除不再有用的设计及代码组件。
- (12) 管理对系统做出的变化。

此外，维护团队的成员与用户、操作员及客户要协同工作。首先，应尽力了解使用用户语言描述的问题；然后，将问题转换成变更请求，变更请求包括对系统现在如何工作、用户希望系统如何工作以及要实现变更所需做出哪些修改的描述。必要的话，一旦设计和代码在修改并测试之后，维护团队应重新培训用户。因此，维护涉及人、软件和硬件的交互配合。

B. 比特币

比特币是一种基于互联网的数字货币，本身通过强大的加密技术来确保没有人复制货币。这使得它可以不依赖于中央政府和管理机构来使用。因此，这种货币对于自由论者、无政府资本主义者以及技术爱好者而言，十分具有吸引力。

这种货币需要很多的计算机来处理和记录交换。这个问题的解决方式是“挖掘”：比特币用户在他们自己的计算机上运行软件来完成所有必要的工作。每当有人成功“完成”工作包（完成难度随着完成人数的增加而上升），他们会收到比特币。这对货币的流通起到了刺激的作用，但同时也吸引了很多正在寻找“不义之财”的掘金者和投机商人，以及那些自认为可以榨取痴迷者的骗子们。

比特币非常有趣的一点是它是完全权力下放的,即不存在中央银行的支持。之前的虚拟货币,例如,E-Gold、Flooz、Beenz、Linden 以及魔兽世界中的金币等,背后通常有组织或企业来控制。没有中央银行的作用意味着,如果政府想针对比特币,他们也找不到攻击目标。

比特币是完全的虚拟货币(即不具备使用价值),但如果当 3 亿用户同时认为它是真实货币时(即可以用于交换),它就变得和美元一样了,可以当作一般的货币使用。

比特币同时存在内在通货紧缩的问题。由于越来越多的比特币被挖掘出来,那么挖掘一个新的比特币需要更多的处理能力。此外,如果钱包文件被删除,那么比特币将永远地消失。

比特币交易的处理由被称为“比特币挖掘者”的服务器端保证其安全。这些服务器在基于互联网的网络上进行通信,并通过将其加入总账的方式确认交易。总账使用 P2P 文件共享技术定期地进行更新和存档。此外,在存档交易的同时,每一个新的总账更新都会产生新的完整的比特币。在每一次更新过程中,新创造出来的比特币的数量会每四年减半一次,直至 2140 年这个数值四舍五入为零。到那时,没有更多的比特币会加入流通,而比特币的总量会达到最大值,即 2100 万。为了容纳这个数量,每个比特币被细分为 8 位小数;组成 1 亿个被称为 satoshis 的小单位。

巴贝奇在《经济学人》很认真的分析了比特币(至少尽力不去取笑它)并认为这很有趣。其他经济学家则引用了其固有的设计问题,来批判比特币(甚至到了称之为骗局的地步)。Paul Krugman 忍住不去取笑它,而是将它视为针对上述存在的经济学问题的一种金本位制的复现。大约 25% 的欧洲中央银行报告的“虚拟货币计划”是关于比特币的。

可以通过比特币来购买真实货物!这些货物通常是互联网服务、极客玩具、违法药物当然还有比特币挖掘硬件。同样,可以买到如啤酒等有用的东西。为了允许诸如比特币极不稳定货币的支付,商家通常会用通用货币来标价,但也可通过当前的市场汇率来接受比特币支付。

第三部分 模拟写作

用户指南

用户指南用通俗的语言介绍如何使用软件,通常与其他有关文档放在一起,如系统管理指南。用户指南通常是用户在需要阅读某些东西时第一个要找的文档,人们通常是在沮丧并对该软件失去耐心的情况下匆匆阅读该指南。

顾名思义,用户指南是为帮助用户理解应用程序或系统而编写的。在撰写用户指南时,应使用具有短句的简单语言,这种风格能帮助用户贯穿整个应用程序。用户指南通常是为非技术人员而编写的,其中的内容和术语的级别应当明显不同于详细复杂的系统管理指南这样的文档。

确定读者

开始之前,首先要确定目标读者、他们的技术知识水平以及他们会如何使用该指南。

在计划过程中,应详尽阐述能确定用户、系统和任务的读者的定义。

软件是用来完成某项特定任务的。用户想要知道该软件可以为他们做什么,如如何打印一页风景画。用户指南要教给用户该软件如何帮助他们完成这些特定任务。

编写用户指南的各个部分

以下各节将介绍用户指南的开头、正文和结尾部分的专门考虑事项。

(1) 开头部分:包括封面、目录,可能还有前言。

- **封面和扉页:**如果该文件是受版权保护的,则应当包括版权声明,如“版权© 2013 公司的名称”放在封面(还有扉页)上。
- **免责声明:**在封面中包含一个标准的免责声明。
- **前言:**使用该章节提及与该软件相关的文档。通常的做法是,包含一个关于“如何使用该手册”的章节作为导言。
- **目录:**要有一个目录表,并在文档结尾处包含一个索引。

(2) 正文部分:在正文部分,应当将流程和参考材料区别并分开。

- **流程:**流程能帮助用户完成具体的任务,可能包括:
 - ✓ 何时、为何,以及如何执行任务;
 - ✓ 执行任务后屏幕上将显示什么内容;
 - ✓ 任务和程序操作的例子。

流程涉及:

1. 列出主要任务。
2. 把每个主要任务分解成子任务。
3. 通过一系列步骤引导用户完成每个子任务。
4. 使用“如果-那么”方法解释用户可做出的决定。

➤ **学会分解:**将任务和信息分成一些小的部分即是所谓的“分解”。

在用户指南中,信息可被分为菜单项和相应结果。将主题和要执行的子任务进行分解,每次分解可以形成手册中新的一章或一节。在编写流程时,对每步进行编号。如果流程内容较少,则以空格分开。各章节应使用一致的结构格式。例如:

- ✓ 以所要执行活动的概述作为每章节的开始;
 - ✓ 描述输入输出;
 - ✓ 提供完成任务的流程。
- **使用“如果-那么”方法:**当用户必须做出选择时,使用“如果-那么”可提供一个清晰的选择和相应结果。对于复杂流程可使用图表。
- **参考材料:**当读者就某一专题需要获取更多信息时会查阅参考材料。参考材料可以包括:
- ✓ 程序选项;
 - ✓ 热键字典;
 - ✓ 错误信息;
 - ✓ 疑难问题解答表;
 - ✓ 常见问题;

✓ 输入输出例子。

- **结尾部分：**术语表和索引（通常是放在步骤之后）是用户指南中两个最重要的附件。
 - **术语表：**当创造新的、程序特定的术语，或以特殊方式使用常用术语时，术语表是非常重要的。
 - ✓ 短的术语表可以出现在目录页之前；
 - ✓ 长的完整的术语表应当出现在末尾；
 - ✓ 术语第一次在文中出现时应当标明。
 - **索引：**对于任何一份超过 20 页的指南，使用索引是十分有用的。没有索引的大型文档不太可能被有效地使用。

创建标准

- **格式：**有些客户可能会看重格式。这一点应在计划时与客户确认。好的格式可以反映该指南的结构，使用清晰的标志更加容易地找到参考信息。

想要做到这一点，可使用如下策略：

 - ✓ 使用标题来组织信息；
 - ✓ 在每页的页眉和页脚处放置章节标题和页码；
 - ✓ 适度地使用章节编号，一种替代方案是，根据读者所要执行的活动，将说明划分成便于管理的小段，为每个新划分出来的小段在右侧页面提供新的章节；
 - ✓ 双栏格式可以通过将标题置于左栏、正文置于右栏的方式被有效利用。
- **风格：**使用合适的风格。这取决于所使用语言的技术水平、引导用户的方式以及所需要遵守的常规惯例。
- **技术语言：**技术语言的水平要与读者所掌握的水平相匹配。宁可低估读者所掌握的知识，也不要高估。尽量减少用户所见技术术语的数量。如果必须定义大量的术语，请在文中使用术语表补充说明。
- **引导用户：**编写流程时，使用主动语态（如单击这里）并直接引导用户（用“你”而非“用户”）。在解释动作时，使用动词的“命令”形式。

PST 修复工具使用指南

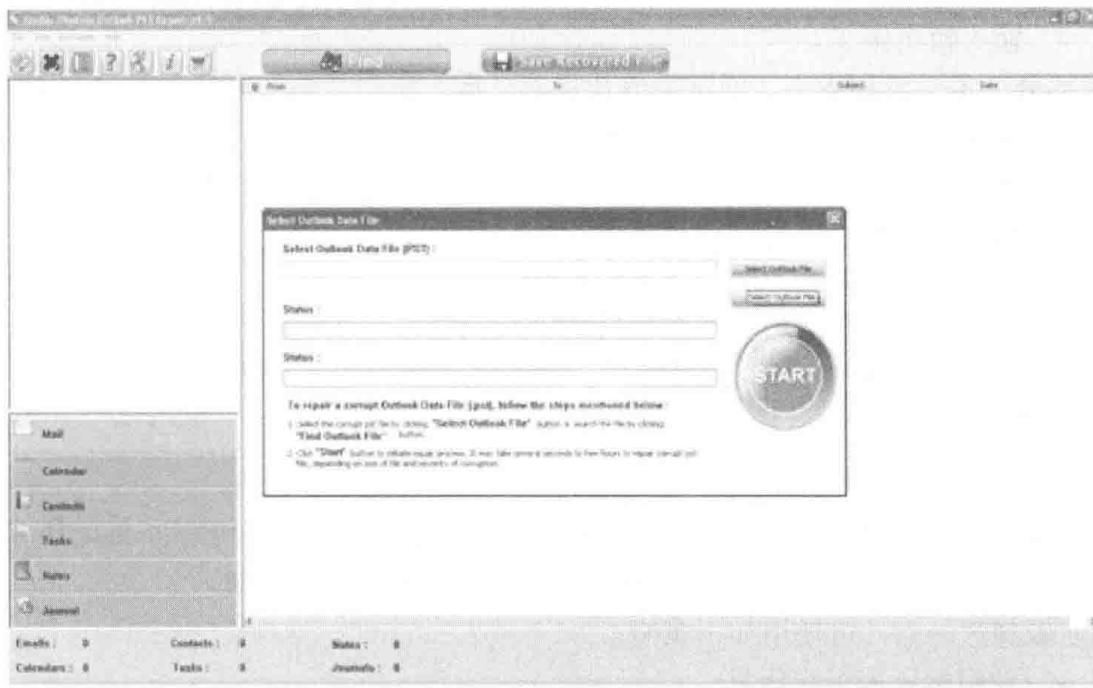
PST 修复工具常用于开发与恢复数据。它提供了自我描述与图形接口，非技术人员能够很容易理解它的工作过程。如果需要恢复很大的文件，它会将数据分成很多小的数据然后逐个恢复。它同样可以通过移除密码恢复那些被密码保护的文件。

为了避免数据损失，可以在微软的 Outlook 中使用 PST 修复工具。它和多个不同版本的微软 Outlook 兼容，如 Outlook98/2000/2003/2007/2010。确保在关闭系统时使用正确的方法避免数据丢失，并且保持最新的 PST 文件。如果数据发生了任何问题，请使用 PST 修复工具恢复数据，避免数据丢失。

无须担心如何使用，下面展示了使用该软件的步骤。

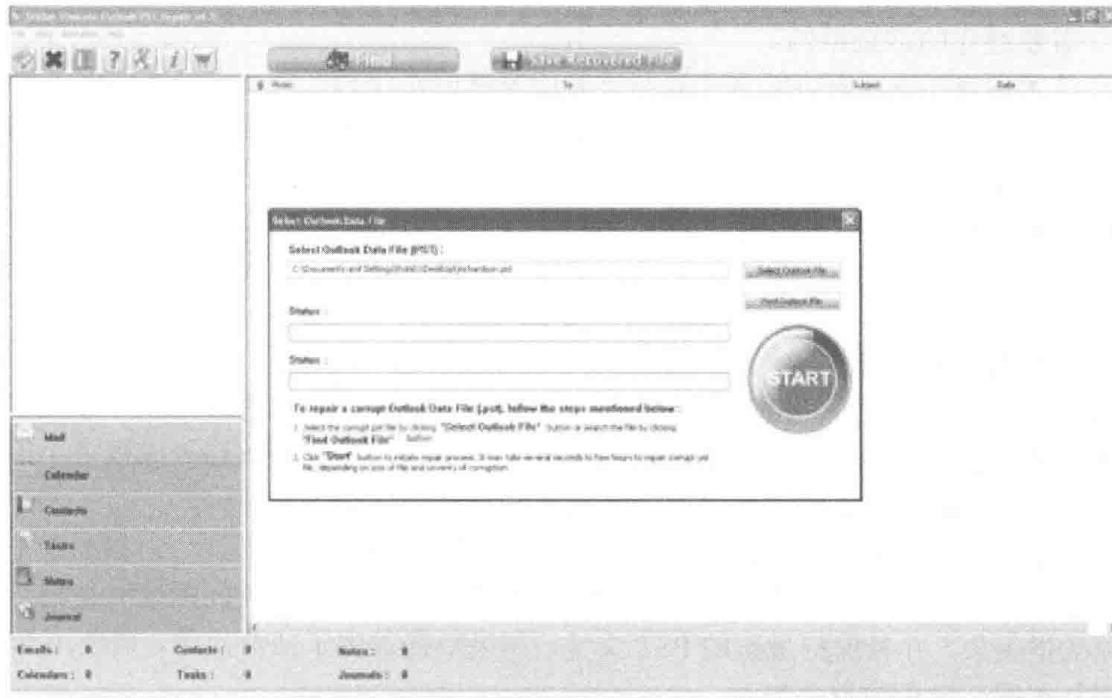
第一步：

首先打开软件，单击 Select Outlook File 选项，选择需要从分区恢复的 PST 文件。



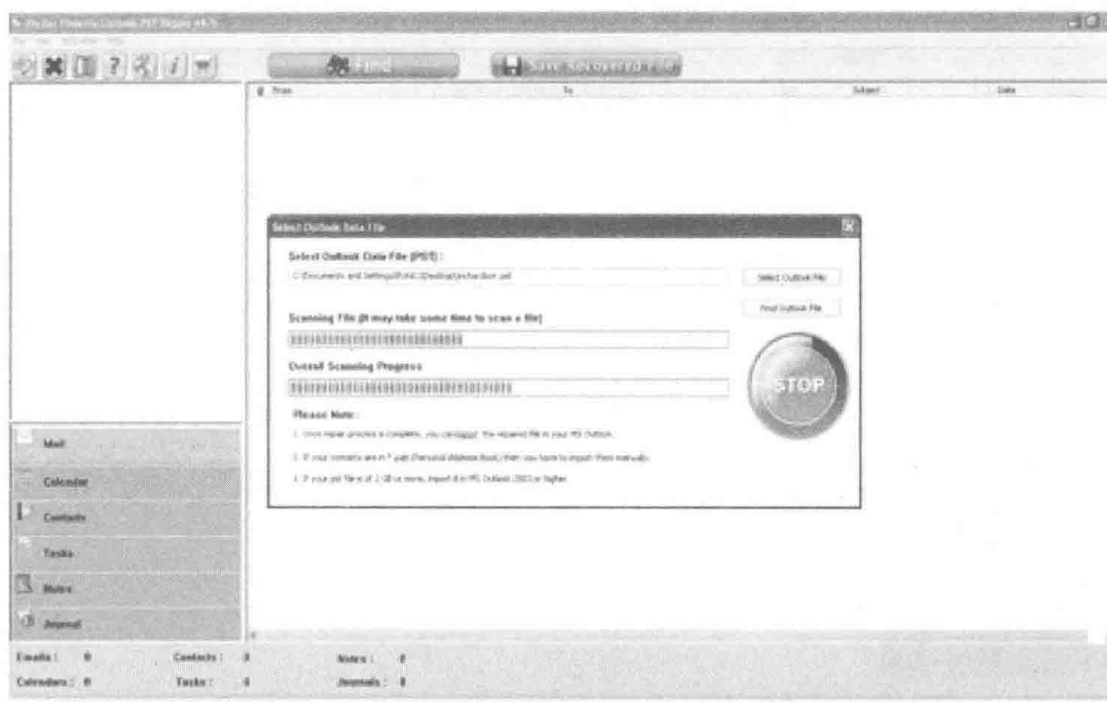
第二步:

现在可以从 Select Outlook Data File (PST)中看到被选中的文件。单击 Start 按钮, 将会开始搜索过程。



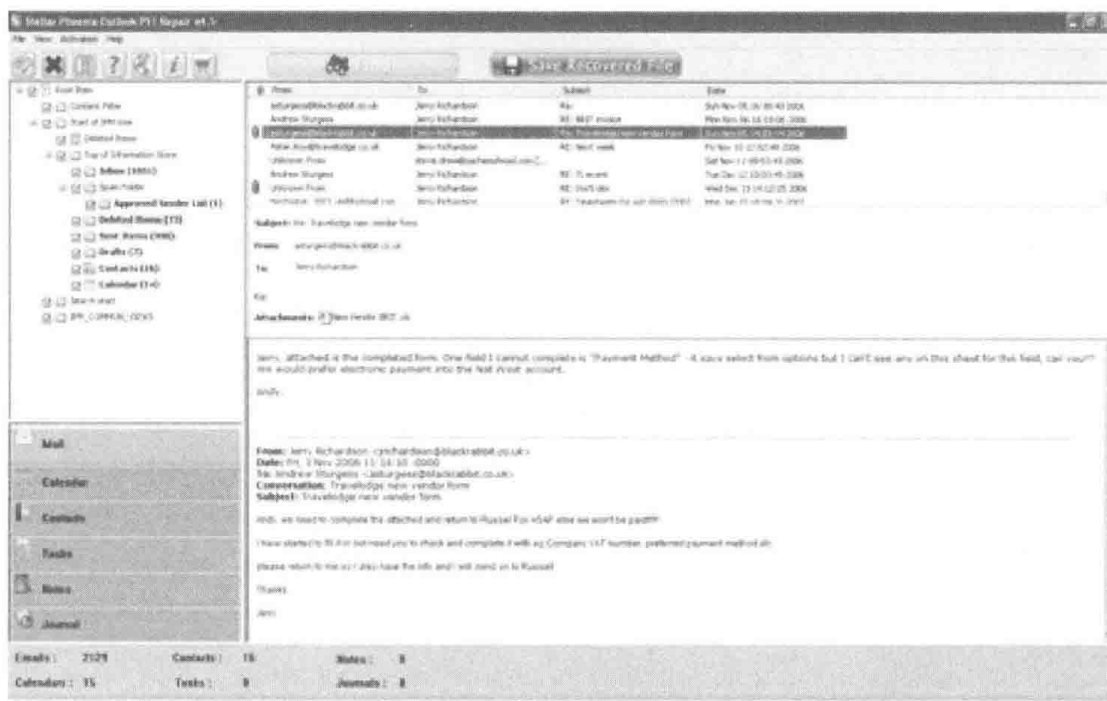
第三步:

屏幕上的进度条将会反应搜索的进程, 同时它还提供了 Stop 按钮以终止搜索进程。



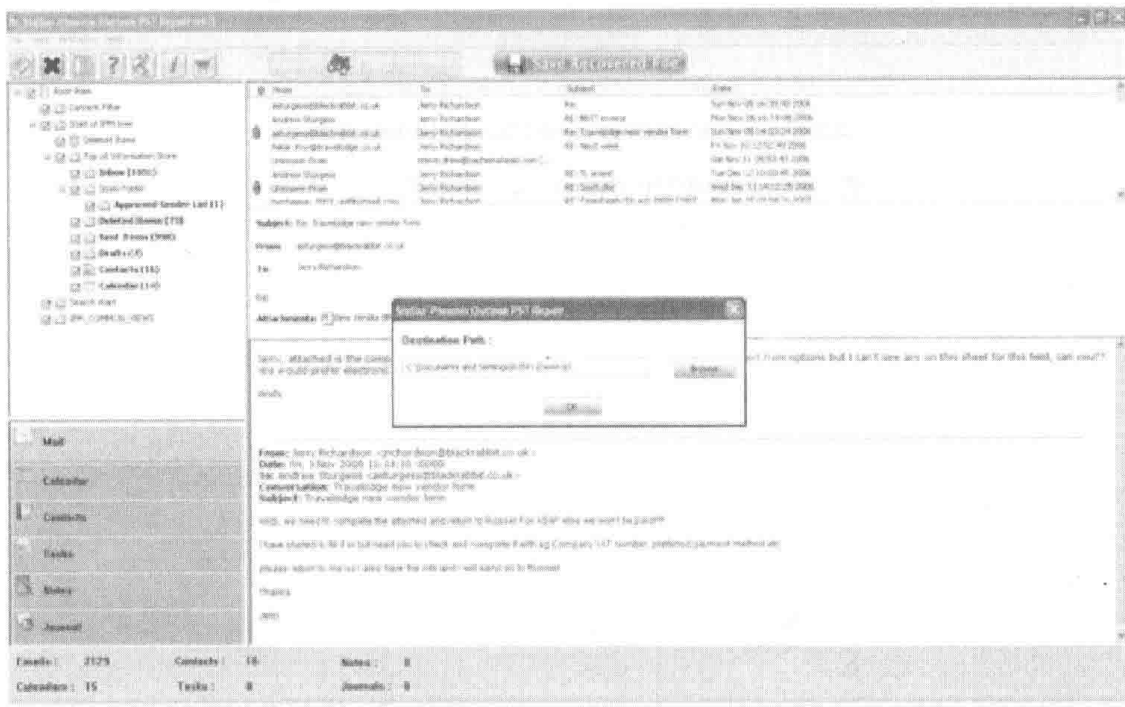
第四步:

搜索完成后，将会在图形接口左边的面板看到所有恢复的数据的列表。这里软件允许用户提前查看所有的邮件消息、联系人、任务、日志等。



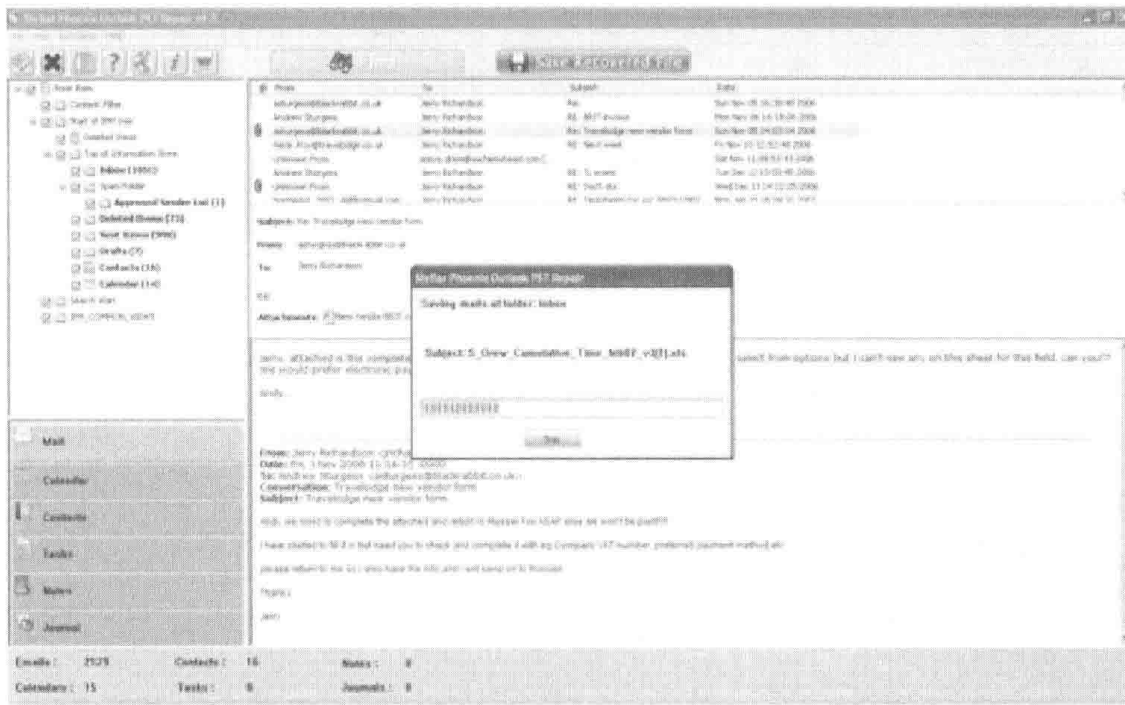
第五步:

单击 Save Recovered File 按钮之后，开始恢复过程，此时可以选择待恢复数据的路径。



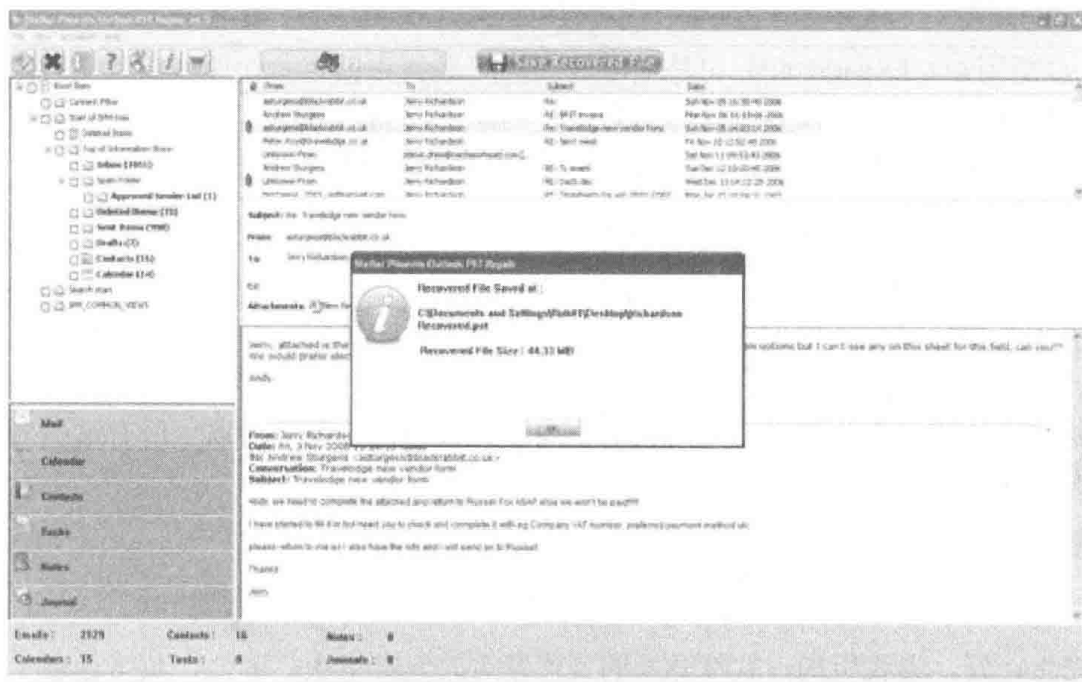
第六步:

软件会在指定的位置保存所有恢复的 PST 数据。



第七步:

在保存过程完成之后，可以看到一个对话框提示 Destination Part 和 Recovered File Size。



第九单元 参加面试

第一部分 听与说

A. 对话：面试

(Kevin 对 Expansion 公司技术部门程序员职位的申请已经得到公司人力资源部门的回复。今天, Kevin 来到公司进行面试)

Smith 先生: 早上好, 欢迎申请我们公司的空缺职位。我叫 Smith, 是人力资源部经理。请问怎么称呼你?

Kevin: 早上好, Smith 先生。我叫 Kevin 李, 我是来申请技术部门程序员职位的。

Smith 先生: 好的, 现在我想了解一下你的情况, 能介绍一下你自己吗?

Kevin: 我是一个积极负责的人, 即使在追求目标的途中会碰上无数的障碍挫折, 我也不会放弃对成功的追求。我喜欢团队工作, 因为我相信, 为获得最好的结果, 分享彼此的意见和主张是很重要的。我对科学技术的最新进展很感兴趣, 并且具备很好的理论背景和动手能力。我热爱 IT 行业, 并希望从事 IT 职业。

Smith 先生: 能不能谈谈你的教育背景?

Kevin: 我于 2013 年 7 月从北京航空航天大学软件学院毕业, 并获得了软件工程的学士学位。

Smith 先生: 你为什么想申请我们公司的职位呢?

Kevin: 众所周知, Expansion 公司是中国成立时间最长和领先的软件外包产品、服务和解决方案提供商之一, 并且具有世界知名度。因此, 我相信在这里工作可以提升我的专业能力, 而且我认为我也可以为公司的发展做出自己的贡献。

Smith 先生: 你是否曾参加过能帮助你准备这类工作的实习呢?

Kevin: 是的。在本科的时候, 作为程序员, 我曾在我们学校与微软公司合作的实验室中, 参加了几个实际项目。作为项目经理, 我曾参与了为四季酒店开发的管理信息系统和为 Giant 公司财务部开发的管理信息系统。此外, 我还为国内一家领先的 IT 公司做过关于全球软件市场的调查和分析, 并用英语完成了商业报告。我认为以上所有这些对于我准备这份工作都是非常宝贵的实习经历。

Smith 先生: 好, 听起来不错! 关于工作你还有什么想要知道的吗?

Kevin: 我想知道, 您认为要从事这个职位还需要什么其他必要条件呢?

Smith 先生: 好, 据我所知, 软件开发的工作量有时候会很大, 因此, 负责这项工作的人可能必须承担很大的压力, 而且必须精力充沛, 并很能干。

Kevin: 是的, 我完全同意您的意见, 在这一点上, 我的实习经历已经让我习惯了这种工作环境。

Smith 先生: 好的, 非常感谢你对我们公司感兴趣。如果我们决定让你加入我们的团队, 你将会在 5 个工作日内收到我们的邮件。

Kevin: 非常感谢您! 期待您的回复。

B. 听力理解: Expansion 公司

作为职位邀请广告的一部分, 这是一份有关 Expansion 公司的概览——一个在 IT 行业提供软件产品及服务的公司。

Expansion 公司是中国著名的软件产品、服务及解决方案供应商之一。Expansion 公司成立于 1990 年, 现已成为中国 IT 行业的领头企业, 在国际上也具有一定的影响。它的总部设立在中国硅谷的心脏——北京中关村。Expansion 公司向全世界提供外包服务, 它的目标是通过向全球顾客提供高质量的服务成为世界领先的 IT 外包公司。

在公司成立的早期, 其主要业务是软件测试服务。随着公司的发展, 如今已能提供越来越多的产品及服务。其中, 软件开发和应用维护服务已成为其关键业务。此外, BPO (业务流程外包) 是一项 Expansion 公司正在开始尝试的业务。

在过去的十年中, Expansion 公司作为包括微软、惠普、IBM 等公司在内世界 500 强的 IT 合作伙伴, 有着良好的业绩记录。日本是目前公司海外市场的主要客户。同时, 通过建立办公场所, 欧洲和美国也将很快成为另外两个巨大市场。

Expansion 公司拥有一支具有丰富经验的管理团队及大量具备良好教育和培训及丰富经验的工程师和项目经理, 而且其员工在 2014 年突破 900 人。通过全体员工的努力, Expansion 公司自成立以来利润一直保持超过 10% 的年增长幅度。

未来, Expansion 公司将不断吸引新的客户来扩大国际影响力, 通过在员工培训以加强经验方面的投入来加强管理团队和人力资源, 并通过在工作中的高效和优良品质来变得更具有经济效益。通过这些努力, Expansion 公司将非常有希望实现它的战略目标。

C. 听写: 软件配置管理

当编写计算机软件时, 会发生变化。正因为变化的发生, 才需要对其进行有效的控制。软件配置管理, 简称为 SCM, 是一系列用来控制变化的活动, 是通过识别很可能发生变化的产品并建立它们之间的联系、为管理不同版本的工作产品定义机制、控制那些强加的变化以及审查和报告所发生的变化等方式。软件配置管理的另一个名字是“变更控制”。

Steve McConnell 在他的《代码大全》一书中描述到: “如果不控制需求的变化, 那么最终很可能会为系统中那些最后被去除的部分写代码。所写的代码可能与系统新的部分不兼容。可能到了最终集成时才发现众多不兼容的地方, 这可能会导致大家互相指责, 因为没有人知道到底是怎么回事。”

因为现在的编程项目如此易变, 所以 SCM 对程序员尤其有用。它所关注的是程序需求、源代码、文档以及测试数据。

软件配置管理系统基于一个简单的概念, 即文件的最终副本保存在一个中央存储库中。程序员从存储库中提取代码文件的副本, 在其上进行工作, 完成后将其提交回中央存储库。软件配置管理系统管理并追踪每个人对版本的修改。所有的软件配置管理系统都具有以下几种基本的特性: 并发管理、版本化和同步。

第二部分 阅读与翻译

A. 计算机专业人员的职业生涯

1999 年, 美国国会对 (美国) 公平劳动标准法案做了修正, 从本质上将计算机专业人员定义为, 其主要职业涉及计算机硬件或软件的设计、配置、分析、开发、修改、测试或安全的任何人。

很多计算机专业人员在信息系统部门工作——负责计算机、数据、软件及支持服务的企业或组织的部门。信息系统部门同时也负责对组织的信息需求进行优先级排序、在需要时修改旧系统以及创建新系统。大多数信息系统部门由首席信息官 (CIO) 或总监领导。CIO 领导若干等级的计算机专业人员——其组织形式可能如图 1 所示。

通常, 以下描述强调了不同信息系统部门职务的责任与相关技能。

系统分析员调研企业或组织及其雇员和客户的需求, 以计划和实现新的或改进的计算机服务。这个工作需要发现问题和研究技术解决方案的能力。好的沟通技能对于与经理及其他

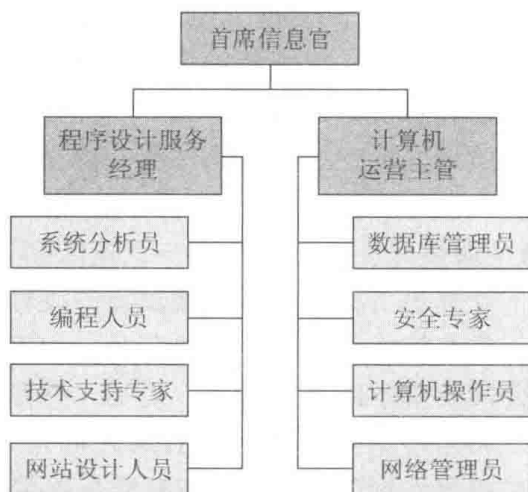


图 1 该组织结构通常在中型企业中使用

雇员的互动非常重要。

计算机程序员（有时也称为程序员、分析员）设计、编码和测试程序。此外，程序员修改现有程序以满足新的需求或消除 bug。计算机程序设计要求专心致志和对于与一个程序设计项目有关的无数细节的好记忆力。

安全专家分析计算机系统对于来自病毒、蠕虫、未授权访问及物理破坏威胁的弱点。安全专家安装并设置防火墙和杀毒软件。他们也与管理人員和雇员一起工作，以制定政策和流程来保护计算机设备和数据。安全专家必须拥有广泛的计算机知识以及能够应用于对所出现的任何危机的快速解决方案的通信协议的知识。

数据库管理员分析公司的数据以决定最有效的收集及存储方法。数据库管理员创建数据库、数据录入表格和报表。他们也定义备份程序，提供用户访问权限，并管理日常的数据库使用。

网络专家/管理员规划、安装并维护一个或多个局域网。这些专家也向被许可的用户提供网络账户及访问权限。他们解决连接问题及响应来自网络用户对于新软件的请求。网络专家/管理员可能要负责维护网络的安全，此外，他们还经常担当网站管理员的职责来维护组织的网站。

计算机操作员通常操作小型机、大型机和超级计算机。计算机操作员监控计算机运行情况、安装软件补丁及升级包、必要时进行备份和恢复数据。

技术支持专家解决软件和硬件问题。这份工作需要好的人际沟通能力和耐心。

网站设计者创建、测试、发布和修改网页。这份工作需要有好的设计意识和艺术天分，以及对人们如何使用图形化用户界面的理解。对 Web 工具的熟悉，诸如 HTML、XML、Javascript 以及 ActiveX，对于这份工作变得越来越重要，对计算机程序设计和数据库管理的知识也是如此。

技术文献编写员为大型编程项目编写文档并撰写在线或打印版的伴随计算机、外围设备和软件的用户手册。有的技术文献编写员为计算机杂志工作，编写关于最新软硬件产品和自动化商业解决方案的专栏。好的写作和沟通能力对于这份工作是很有用的，快速学习如何使用新计算机和软件的能力也同样重要。

计算机销售人员或销售代表销售计算机或 IT 服务。销售代表可能会对潜在的公司客户进行访问或是作为邮购计算机公司订货处的工作人员。销售代表的起薪往往很低但通常通过佣金来补偿。得力的销售代表具备良好的交际能力、能牢记技术说明的能力以及对于业务问题和解决方法的理解力。

质量保证专家参与软件的 α 测试和 β 测试周期，寻找 bug 或其他可用性问题。这份工作有时候是指流水线上检查并测试芯片、电路板、计算机及外围设备的工人。一名得力的 QA（质量保证）专家具有一双追求细节的慧眼和追求完美的激情。

计算机工程师设计和测试新的硬件产品，如计算机芯片、电路板、计算机及外围设备。

生产技术员参与计算机芯片、电路板、系统单元或外围设备的制造。这些工作中的一部分需要基本的技工能力，而另一些则需要经过微光刻方面的专门训练。

美国信息技术协会（ITAA）将 IT 职位分为 8 大类。其中一些职位名称列出如下：

数据库管理和开发：包括数据库管理员、数据库分析员、数据库开发员、数据库经理、数据库安全专家、DSS（决策支持服务）以及知识架构师。

数字媒体：包括二维/三维设计师、动画专家、音频/视频工程师、图案设计师（平面设计师）、媒体专家、媒体/教学设计师、多媒体作家、多媒体制作专家、多媒体开发人员、多媒体专家、制作人、流媒体专家以及虚拟现实专家。

企业系统分析和集成：包括应用集成师、业务连续性分析师、跨企业集成师、数据系统设计师、数据系统经理、电子业务专家、信息系统架构师、信息系统规划师、系统分析员、系统架构师以及系统集成师。

网络设计和管理：包括通信分析师、网络管理员、网络分析员、网络架构师、网络工程师、网络经理、网络运营分析员、网络安全分析员、网络专家以及网络技术员。

编程/软件工程：包括应用分析员、应用工程师、业务分析员、计算机工程师、操作系统设计员/工程师、操作系统程序员/分析员、编程经理、程序员、程序员/分析员、软件应用专家、软件架构师、软件设计工程师、软件工程师、软件质量保证专家以及软件测试师。

技术支持：包括呼叫中心支持代表、客户联络人、客户服务代表、客户支持专员、用户服务部专家、用户服务部技术员、PC 支持专家、PC 系统协调员、产品支持工程师、销售支持技术员、技术客户经理、技术支持工程师以及技术支持代表。

技术写作：包括桌面出版人、文档专业人员、文件专业人员、编辑、电子出版专家、电子出版商、教育设计人员、在线出版人员、技术通信员、技术编辑、技术出版物经理、技术作家。

网站开发和管理：包括网络管理员、网络架构师、网络设计师、网页开发人员、网站开发人员、网络专家以及网站管理员。

B. 三维打印

你可能从新闻播报员和记者那里听说过三维（3D）打印技术并惊讶于他们的描述。这种机器让人联想到星际迷航中可以凭空创造物体的复制机。它可以使用玻璃、金属、尼龙以及超过百种材料进行“打印”。它既可以被用于制作无意义的小模型，比如打印尤达的模型，同时它也可以用于制作工业原型、最终用户产品、准合法的枪支、飞机发动机甚至使用人们自己的细胞来制作器官。

我们生活在一个可以见证“第三次工业革命”的时代。3D 打印技术，或更专业地说添加剂制造技术，让我们摆脱了亨利福特时期的大规模生产线，进入到新的可定制、一次性生产的时代。

当需要洗衣机的部分零件时，在这个时代，订购的零件或许来自于一个用非常昂贵的模具注塑而成的一起批量生产的成千个零件，通过船运从中国运至分销商，分销商再交给修理工，最后到达用户的手中。在未来，可以轻松的使用 3D 打印技术按照下载的 CAD 文件在家里制作零件。如果没有合适的打印机，也可以去当地的代工厂来打印。

3D 打印机使用大量不同类型的添加剂制造技术，但是它们都有一个共同点：它们通过逐层的建立来创造三维物体，直至整个物件完成制作。这就像是在一张纸上打印二维文件，只不过是增加了第三个维度：高度，即 Z 轴。

每一个打印层都是实际物体的水平截面薄片。试想一个多层蛋糕，面包师每次放置一层直到整个蛋糕制作完成。3D 打印技术也有点类似，只不过比制作蛋糕更加精细。

尽管大多数人还没听过 3D 打印，但实际上这项技术已经被投入使用长达数十年。制

造商在设计阶段使用这种打印机来制作传统制造业的原型。但是直到最近几年为止, 这项技术一直是昂贵和低效的。

现在, 只需要花费数万美元就能购置的 3D 打印机能为公司在成型过程中节省数倍的成本。例如, 耐克公司使用 3D 打印机来制作多种颜色的原型鞋, 他们曾经需要在每一个原型上花费数千美元而且需要等待数周的时间。现在, 他们只需要花费数百美元, 并且可以在计算机上立即修改后当天就拿到重新打印的模型。

一些公司使用 3D 打印机来满足短期制作或客户定制的需求, 在这些情况下打印结果不是原型而是真正的最终用户产品。随着 3D 打印技术的速度提升以及成本下降, 这种情况会越来越多, 并且可以预见到更多定制产品将会出现。

即使没有设计自己的 3D 模型, 仍然可以打印非常有意思的物件。可以从诸如打印服务平台 Thingiverse、3D 部件数据库以及 3D 仓库等模型仓库中免费下载相关模型文件。

人们都可以打印什么? 这是没有限制的。一些人打印珠宝首饰, 一些人打印洗碗机等设备的备用零件, 一些人专门发明各种原创的东西, 一些人创造艺术品还有些人为他们的孩子制作玩具。当类如金属、塑料、玻璃以及其他材料(甚至金银)可用时, 几乎所有的东西都可以打印。

这是一种在能源利用、废物处理、定制、产品可用性、艺术、医药、建筑、科学技术以及生产等方面的颠覆性技术。它将会改变这个世界。

第三部分 模拟写作

简历

简历是重要的求职工具, 如果我们经验丰富, 就应该知道最好将简历长度限制在 1~2 页。因为简历向潜在雇主展示第一印象, 因此要花时间以确保简历有良好的组织和设计, 易于阅读且没有错误。只选用高质量的纸张。

分析背景

确定想寻找什么类型的工作, 并确定关于背景的哪些信息会是最重要的且是与该类工作相关的。列出如下内容:

- 所在学校、学位、专业、学术荣誉、平均成绩(如果是最近毕业)、特别的及与该工作有关的学术项目;
- 工作经历、在每份工作主要和次要职责、每份工作何时开始以及做了多长时间、职位晋升和在工作中所获得的技能;
- 对于所申请的该类型工作有价值的其他经验和技能, 有助于增加学习经验的课外活动, 所具备的领导才能、人际关系和沟通能力, 所完成的任何团队合作工作, 所具备的计算机技能。

组织简历

大多数简历根据以下分类按时间顺序组织信息:

- 标题(姓名和联系信息);
- 求职意向(可选);

- 教育背景;
- 工作经历;
- 计算机技能 (可选);
- 荣誉和活动;
- 推荐人。

如果是最近毕业且没有什么工作经验,则首先要列出教育背景。如果有多年的工作经验,包括与当前所申请职位直接相关的工作,则首先列出工作经历。在教育背景和工作经历的章节中,使用时间顺序,最近的经历放在第一位,次于近期的放在第二位,以此类推。

标题

在简历顶部,写出所有的联络信息,包括姓名、地址、电话、传真和电子邮件地址。确保姓名在该页上足够突出。如果只有一个地址,可把它放在该页上方正中间的位置。如果有两个地址,如学校的和家庭的或是临时的和永久的地址,把学校的或临时的地址放在左边,家庭的或永久的地址放在右边。把地址都放在姓名的下方。

求职意向

求职意向引出简历中的材料,并帮助读者快速了解目标。书写求职意向最多不要超过三行。例如,旨在解决工程问题的全职软件程序员/分析员。

教育背景

列出所就读的学校、所获得的学位及获得时间、专业和所获得的学术荣誉。如果平均成绩是 3.0 或更高,请列出。只有令人印象深刻或是当简历内容不够丰富时,才列出所学课程。

工作经历

可以在该标题下按时间顺序以最近职位开始用倒序方式列出工作经历,或从功能上组织工作经历,通过将相似类型的工作聚集在一个或若干章节中,使用特定标题,如“管理经验”或“主要成就”。

- 包括与所申请的职位直接相关的工作或实习经验;
- 包括课外活动,特别是那些涉及领导职位或社区服务的活动;
- 在每项工作经历下方简述主要和次要职责;
- 准确简明地使用动词,不要使用“我”。

计算机技能

如果申请的职位需要计算机知识,则应包括所熟悉的特定语言、软件和硬件。

荣誉和活动

如果简历中(少于两页)还有空白,也可以写上例如外语熟练程度、写作及编辑能力、专业技术知识、学生或社区活动、职业或俱乐部会员、出版的作品等。要有选择性,不要提供重复的材料,并且只提供与求职意向相关的材料。

推荐人

除非简历太稀松,否则的话不要列出推荐人。面试时,打印一份推荐人列表并随身携带。列表应包括主要标题“[姓名]的推荐人”。

第十单元 开始工作

第一部分 听 与 说

A. 对话：开始工作

(祝贺 Kevin 成为 Expansion 公司信息系统部的一名新程序员！今天是他第一天工作，他将认识一些新同事并迎来一个全新的工作环境)

Kevin: 很高兴见到你, John 先生, 我叫 Kevin, 我很高兴能来上班。

John 先生 (公司首席技术官): 欢迎你来到我们公司, Kevin。你将在信息系统部工作。那么, 现在请让我向你介绍一下我们的同事。

(Kevin 跟着 John 先生走进了办公室)

John 先生: 大家好, 今天我很高兴向大家介绍一位新同事, 他叫 Kevin 李, 从今天起, 将与大家一起工作。

Kevin: 大家好。

A, B, C: 你好, 李先生, 欢迎你!

Kevin: 谢谢, 我刚刚毕业, 这是我第一次在跨国公司工作, 希望大家能帮我适应从学校生活到职业工作环境的过渡。

A: 没问题, 李先生, 我是 Jenny, 负责办公室工作。这是你的办公桌。

Kevin: 非常感谢, Jenny 小姐。

A: 叫我 Jenny 就可以了。公司中每个人都直接称呼名字, 这是自公司成立以来就已经形成的传统了。

Kevin: 好的, 我尽量。

B 和 C: 很高兴认识你, 相信我们将在以后的工作中合作愉快!

Kevin: 我也是。

John 先生: Kevin, 你如果有什么问题的话尽管问, 大家都很友好。还有其他问题吗?

Kevin: 好的。对不起, 顺便问一下, 你能告诉我, 我的具体工作和职责是什么吗?

John 先生: 这周大部分时间你用来熟悉办公室的工作, 并阅读下周将要加入项目的相关文档。还有其他问题吗?

Kevin: 没有了, 相信以后可能还会有很多问题, 谢谢。

(工作了一整天, 现在是下班的时间了)

Jenny: Kevin, 已经工作了快一天了, 你觉得工作怎么样?

Kevin: 不错, 谢谢你的帮助。一整天都很忙, 现在觉得有点累。

Jenny: 我第一次在这里工作的时候, 也有同样的感受。不过一段时间以后, 就感觉好多了, 我相信你会很快适应的。

Kevin: 嗯, 我觉得工作效率很高, 而且也很有挑战性。这里的每个人工作都很努力并且很合作, 大家都具备很强的能力和专业技能。我真的很喜欢这样的工作环境。

B. 听力理解: 公司的组织结构

对于新成员来说, 尽快了解其公司的组织结构及成员关系非常重要。以下关于公司组织结构的简要描述将有助于新雇员了解他(她)的公司。

在公司中, 管理者及其下属组成了一个组织。有一些“首脑”作为领导。首先是 CEO (首席执行官), 他是负责公司整体运作的执行者, 通常是公司的总裁或董事长。此外, 首席财政官(CFO)是负责监督整个公司财务活动, 包括财务计划、现金流动等的高级经理。另外一个高级经理是首席技术官(CTO), 负责公司中与技术相关的部门, 如 IT 支持部及信息系统部。首席运营官(COO)是负责管理公司日常运营并且向 CEO 报告的高级经理。他们组成了公司的高级管理层。

上述组织结构是按等级进行分类的, 因此在公司中具有层次结构: CEO 是公司的领导者; CTO、CFO 和 COO 直接向 CEO 汇报; 市场总监和客户服务经理受 COO 领导。随着公司的发展, 公司的组织可能会变得更大, 而实际的结构很可能会更加复杂。也就是说, 这种结构中的每个位置都是可扩展的, 可能 CTO 和 CFO 各自都拥有其下工作的团队。在一些公司中, 还有首席商业扩展和营销官(CBDMO)和首席战略官(CSO)等。

C. 听写: 开源运动

在计算机发展的早期, 软件是捆绑在计算机上的, 包括软件的源代码。程序员调整和修改这些程序, 并且乐于将其所做的改进进行分享。在 20 世纪 70 年代, 各个公司开始保留源代码, 并且软件也成为了一块很大的业务。

与将代码保密相反, “开源”意味着代码可以被编写它的人免费分发, 利用分布的同行评审的力量和过程的透明度。开源的承诺是, 更高的质量、更高的可靠性、更多的灵活性、更低的成本以及终结损人利己的商家闲置代码。

20 世纪 60 年代, 有权访问阿帕网(ARPANET)的研究人员使用称为“征求评议”的过程来制定电信网络协议。以同时期的开源代码作为特色, 这项合作性的行动导致了 1969 年因特网的诞生。在免费软件运动中使用“开源”一词的决定始于在加利福尼亚州的帕洛阿尔托召开的一次战略会议上, 作为对 Netscape 公司 1998 年 1 月对 Navigator 浏览器源代码发布的公告的回应。

到目前为止, 大多数推动因特网、创建操作系统和生产软件的革新的程序设计都是开源代码的结果。开放源代码已经促进了计算机技术的高度发展, 包括最有名的 Linux 和 Apache, 这两者在市场上给微软公司带来了巨大的挑战。

Steven Weber 在其《开源的成功》一书中提出, 开源的成功并不是不符合经济规律。开源社区也有标准、规则、决策制定规程和认可机制作为指导。

第二部分 阅读与翻译

A. 公司的组织结构

考虑我们所熟悉的任何规模的组织，其目的都是履行一项服务或提供一种产品。如果是非盈利的组织，如它可能提供教育学生的服务或者为饥荒灾民提供粮食。如果是盈利的组织，如它可能销售安装计算机的服务或者计算机产品。信息——不论是否基于计算机——必须在组织内部以一种能够帮助管理者和该组织达到其目标的方式流动。为此，组织通常是水平和垂直的结构——水平地反映职能，而垂直地反应管理级别。

依赖于其所提供的服务或产品，大多数组织具有履行如下 6 项职能的部门：研发（R&D）、生产（或运营）、市场和销售、会计和财务、人力资源（人员）、信息系统（IS），如图 1 所示。

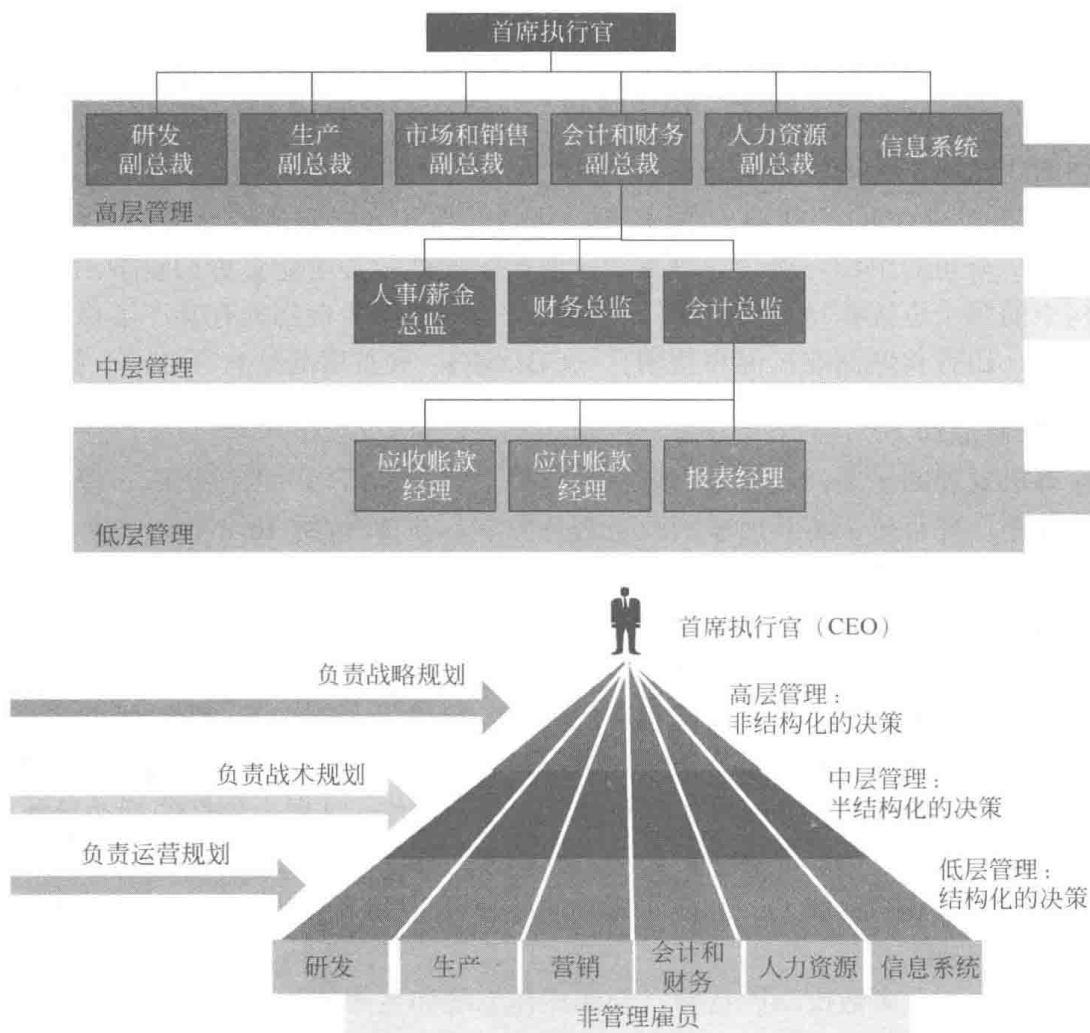


图 1 组织结构图

研发：研发部门负责两项工作。

- (1) 进行基础研究，将研究发现与组织现有的或新产品进行关联。
- (2) 做产品开发，并且测试和修改由研发人员所创建的新产品或服务。

生产(运营):生产部门生产产品或提供服务。在制造公司中,就是通过人或机器把获得的原材料变成成品。在另一种类型的公司中,生产部门可能会管理采购、管理库存和控制商品和服务的流通。

市场和销售:市场部门负责监管广告、促销和销售情况。该部门的人员向顾客或客户规划、定价、宣传、推广、包装和分销服务或产品。

会计和财务:会计和财务部门负责处理所有的财务事项;处理现金管理、支付票据和税收、支付薪水、记录账目、进行投资,并编写财务报表和报告;还负责制定财务预算,并在收到其他部门的信息后预测财务业绩。

人力资源:人力资源(或人事)部门寻找和聘用人员,并管理病假和退休事宜,还关注员工补偿、职业发展、员工关系和政府规章等。

信息系统:信息系统部门管理该组织基于计算机的系统,并计划和购买新系统。

大型组织一般有三个管理层次——战略管理、战术管理和运营管理。这些层次可显示在一个组织结构图上,示意图显示了组织内部员工之间正式关系的层次。三个层次中每个层次的管理者都有不同层次的责任,并因此需要做出各种不同的决定(见图 10-1)。

战略级管理:高层管理者关注长远的或战略的规划和决策。这个高层由 CEO 及另外几个具有如 CFO、COO 和 CIO 等头衔的副总裁或副总经理领导。战略决策是复杂的决策,很少根据预定的例行程序而定;它们需要决策者的主观判断。例如,战略决策涉及如何为增长提供资金和首先应该抓住什么样的新市场。确定公司的 5 年目标,评估未来的财务资源和回应竞争对手行动等也都是战略决策。

战术级管理:战术级或中层管理者,制定战术决策以实现组织的战略目标。战术决策的制定没有一个已明确定义的信息流程的基础;它可能需要详细的分析和计算。战术级管理者包括工厂经理、部门经理、销售经理、分公司经理和人事总监。

运营级管理:运营或低层(督导级)管理者制定运营决策——可以根据清晰定义的例行流程集来制定可预见性决策。这些管理人员将主要关注于监督普通员工,监督日常活动,必要时采取纠正措施。运营级管理者的一个例子是负责清查进货的仓库管理员。

值得注意的是,由于高层的管理人员很少,中低层的管理人员很多,因此这种管理结构类似于一个金字塔。这也是一种层次结构,因为大部分的权利集中在高层。

B. 谷歌眼镜

谷歌眼镜尝试将数据从台式计算机或便携设备(如电话、平板电脑)上解放,直接展现在用户眼前。

本质上来说,谷歌眼镜是一个加装了摄像机、显示器、触摸板、电池以及扩音器的眼镜框,所以可以暂停视角、记录视频、拍照片、搜索及解说眼前的事物。

这个原理在科幻小说已经存在了很多年,但现在它被略显粗笨地实现了。事实上“头部显示器”,即将数据直接展示在眼前,早在 1900 年反射瞄准镜发明时就已经实现了。

谷歌眼镜通过显示技术将数据展现在面前(或至少在右上方)的棱镜屏幕上。这种设计使得用户可以轻松的看到屏幕中的内容,而不会遮挡视线。根据谷歌方面的消息,这种显示相当于 8 英尺外的 25 寸显示器。这里没有标准分辨率的官方信息,但很多人认为是 640×360。

将数据直接展现在眼前可以带来很多好处，而谷歌眼镜已经实现了其中的一部分。方向变得更加直观（尽管现在看来谷歌眼镜中没有加入 GPS，所以需要与手机一起使用）；可以对谈话进行实时翻译和记录；也可以查看和回复信息，所有这些都是在线进行的。

嵌入的摄像机显然不需要取景器，因为它仅仅记录第一人称视角，并且允许对看到的事物进行拍照和录像。

每一个需要看屏幕的功能都会摆在面前。

控制这些数据同样简单巧妙。通过眼镜架上的扩音器和触摸板，可以通过简洁的手势选择想要做什么或通过语音让谷歌眼镜来解析命令。

谷歌眼镜也可以通过骨诱导技术来提供音频。这项技术可以通过振动头骨来产生声音，从而比普通耳机音质更好且方便携带。

与谷歌公司自己列出的特点一样，谷歌眼镜的早期应用程序也对这种设备的潜力进行了发掘。

就像无须解释的观看照片和电影一样，也可以使用谷歌聚会软件来与朋友进行视频会议并向他们展示眼前的事物。

也可以使用谷歌地图来获取方向，尽管可能需要将眼镜与手机相连接以弥补没有内置 GPS 的问题。

为了实现这点，谷歌公司提供了 MyGlass 应用程序。该应用程序可将谷歌眼镜与 Android 手机相匹配。就像分享 GPS 数据一样，这意味着可以通过扩音器以及谷歌的音频文本转换功能来实现信息的接收、查看以及答复。

谷歌公司通过收购 DNNresearch 语音识别公司，大力促进它的眼镜工程。

这个功能也可以将听到的语言转化为自己的母语来显示出来。显而易见的是，当在其他国家时会需要 Wi-Fi 链接或庞大的数据计划来实现这个功能，但如果实现了将会非常有意思。

第三方也已经开发出了一些非常酷、非常惊人的谷歌眼镜应用程序，包括一个可以在人群中找到朋友的功能以及一个允许口述邮件的应用程序。

《纽约时报》的 Android 应用程序就给出了一个如何展现所需新闻信息的样例，包括标题、署名、适当的图片以及文章已经发布的时间。

其他有意思的想法还包括在等飞机时可以看到航班详细信息的应用。基本而言，谷歌眼镜的应用是无限宽广的。

第三部分 模拟写作

商务电子邮件

电子邮件是电子、计算机辅助的在线交流工具。在商务交往中，电子邮件用于传递每日商务行为所需的各种信函：简短的消息、备忘录和信件、复杂的报告、数据表、图和图表、设计图、图片，无所不包。如果信件能够由计算机生成，或是扫描或下载到计算机上，那么它就可以通过网络空间以电子的方式发送到另一台计算机上。

在这个指导中，将会阅读到如何写商务电子邮件和在编排商务电子邮件格式上有用的窍门。每一章节都提供有用的信息，以便在商务交往中更熟练地使用电子邮件进行交流。

商务电子邮件的格式编排

商务电子邮件的格式与商务备忘录相似。事实上,商务备忘录中出现的基本标题元素被编进计算机所生成的每个电子邮件程序的模板中。但是,商务电子邮件的沟通作用却是备忘录所没有的。为了体现电子邮件的多种功能,商务信件中排版元素可以手动加入电子邮件的正文中。

商务电子邮件既可作为内部交流也可作为外部沟通的工具。它的三个主要组成部分是标题、正文和签名区。这要取决于信函的性质,并根据作者的意见,商务电子邮件还可能加入开头称呼语和信尾结语。

商务电子邮件的标题

商务电子邮件的标题包括最多 6 个不同的信息域。它们位于电子邮件模板的顶部,正好位于工具栏的下面。

无论何时单击任一电子邮件程序工具栏中的新邮件按钮、回复按钮、回复所有按钮或转发按钮,模板都将自动显示。

模板中的每个信息域都用来包含特定的信息,它们的定义预先印好在左边的空白处,内容如下:

- 收件人;
- 发件人;
- 抄送;
- 密送;
- 日期;
- 主题;
- 附件。

上述这些信息域中的一些不总是可见的。例如,密送和附件,只有在被发件人激活后才可见,而在有些电子邮件程序中,发件人和日期可能在发件人的模板中并不可见。

商务电子邮件的正文

商务电子邮件的正文与商务信件或备忘录没有区别。唯一格式上的区别是:电子邮件程序自动将正文编排为不隔行打,采取齐头式。这是电子邮件程序的一项设计功能,目的在于确保使收件人看到的电子邮件样式与发件人发送的完全一致。

撰写电子邮件时,原则是采用较短小的段落,特别是当在线读者往往只是粗略浏览正文的时候。而且,许多显示器一次只能显示 20~25 行,所以短一些的段落更适合阅读。此外,很长的段落对眼睛不好,阅读起来也更困难。

写电子邮件时要考虑到收件人的时间;如果电子邮件超过两页,那么最好采用信件或备忘录的形式。作为附件发送,那么收件人在时间允许时,就可以将冗长的信件或备忘录打印出来阅读。

通常来说,商务电子邮件是使用普通文本而不是 HTML 形式来发送的。使用不同的字体、颜色、大小和格式(如斜体或粗体)在使用文字处理软件创建文档时非常有用,但是在电子邮件程序之间进行发送时,就可能会出现问題。

HTML 格式的电子邮件除了可能会增加收件人的下载时间,在不同的计算机上显示也可能不尽相同。如果对这些区别不是很熟悉的话,那么就选择单纯的文本形式,然后继续。

商务电子邮件的签名区

商务电子邮件中签名区的作用与商务信函中标题或信头的作用类似。也就是说,这部分提供的是发件人的联系方式,是电子邮件的最后一部分,通常位于签名线以下左边的空白处。

签名区应该包含收件人可能需要的所有联系信息,以方便回信。开头是发件人的姓名、头衔、单位,随后是地址、电话号码、电子邮件地址以及网址等。

商务电子邮件的开头称呼语和信尾结语

商务电子邮件中开头称呼语和信尾结语的格式规则与商务信函和备忘录相同。

一些正式的表达,如“尊敬的 Ortiz 女士”和“谨致问候”等,适用于写给不熟悉的人的商务电子邮件;另一方面,如果商务电子邮件的作用是备忘录的话,则应当删掉称呼语和信尾结语。多数情况下,称呼语和信尾结语不必很正式,特别是在先前的联系中已经完成了必备的礼仪形式的情况下。

Answers (练习答案)

Unit 1

Part 1 Listening & Speaking

Listening Comprehension

1. C

2. B

3. A

原文如下:

Software Engineering

Today, computer software has become the key element in the evolution of computer-based systems and products and one of the most important technologies on the world stage. Over the past 50 years, software has evolved from a specialized problem solving and information analysis tool to an industry in itself and becomes a dominant factor in the economies of the industrialized world.

Yet we still have trouble developing high-quality software on time and within budget. Why does it take so long to get software finished? Why are development costs so high? Why can't we find all errors before we give the software to our customers? Why do we spend so much time and effort maintaining existing programs? And why do we continue to have difficulty in measuring progress as software being developed and maintained? These questions and many others demonstrate the industry's concern about software and the manner in which it is developed — a concern that has lead to the adoption of software engineering practice.

Then, what is software? And which characteristics of software make it different from other things that human beings build? Software is a logical rather than a physical system element, and some characteristics make software special. First, it is difficult for a customer to specify requirements completely, and difficult for the supplier to understand fully the customer needs as well. In defining and understanding requirements, especially changing requirements, large quantities of information need to be communicated and assimilated continuously. Second, software is seemingly easy to change and is primarily intangible, and much of the process of creating software is also intangible, involving experience, thought and imagination. In addition, it is difficult to test software exhaustively.

Software — programs, data, and documents — addresses a wide range of technology and application areas, yet all software evolves according to a set of laws that have remained the same for over 30 years. The intent of software engineering is to provide a framework and a solution for building higher quality

software. It includes greater emphasis on systematic development, a concentration on finding out the user's requirements, formal/semi formal specifications of the requirements of a system, demonstration of early version of a system (prototype), greater emphases on trying to ensure error free code and so on.

From the time in 1968 when the phrase "software engineering" was first used at a NATO conference until the present day, software has come a long way. But it still has a very long way to go if it is to be considered as mature as other engineering disciplines.

Dictation

- | | | | | |
|------------------|--------------------|---------------------|------------------|----------------|
| 1. known | 2. project manager | 3. Operating System | 4. human | 5. publication |
| 6. bookshelf | 7. statement | 8. manpower | 9. later | 10. law |
| 11. environments | 12. collaborative | 13. individuals | 14. originally | 15. embodied |
| 16. drop | 17. circulation | 18. programming | 19. productivity | 20. run |

Part2 Reading and Translating

Section A

- I. 1. F 2. F 3. T 4. T 5. F
- II. 1. D 2. D 3. B
- III. 1. 由计算机与通信系统的一体化以及复杂的图形用户界面而产生的新技术对软件工程师提出了新的要求。
2. 不同类型系统和使用这些系统的不同类型组织的广泛的多样性, 意味着我们需要软件开发方法的多样性。

Section B

- I. 1. F 2. F 3. T 4. F 5. F
- II. 1. C 2. D 3. D
- III. 1. 这种概念是指用户通过他们的计算机或手持设备使用的基于互联网的服务器提供的, 而不是来自本地设备的应用程序、存储空间以及其他计算资源。
2. 云计算的最大优势主要体现在, 由于数据是在线存储而不是本地存储, 用户可以在任何互联网覆盖地区访问数据, 并且当设备丢失、被盗或损坏时, 数据依然安全。

Unit 2

Part 1 Listening & Speaking

Listening Comprehension

1. A 2. C 3. C

原文如下:

Software Requirements

It is necessary to understand requirements before design and construction of a computer-based system can begin. To accomplish this, a set of requirements engineering tasks are conducted. Requirements engineering occurs during the customer communication and modeling activities that we have defined for the generic software process. Seven distinct requirements engineering functions — inception, elicitation, elaboration, negotiation, specification, validation, and management — are conducted by members of the software team.

At project inception, the developer and the customer as well as other stakeholders establish basic problem requirements, define overriding project constraints, and address major features and functions that must be present for the system to meet its objectives. This information is refined and expanded during elicitation, which is a requirements gathering activity that makes use of facilitated meetings and the development of user scenarios.

Elaboration further expands requirements into an analysis model — a collection of scenario-based, activity-based, class-based, behavioral, and flow-oriented model elements. A variety of modeling notations may be used to create these elements. The model may reference analysis patterns — characteristics of the problem domain that have been seen to reoccur across different applications.

As requirements are identified and the analysis model is created, the software team and other project stakeholders negotiate the priority, availability, and relative cost of each requirement. The intent of this negotiation is to develop a realistic project plan. In addition, each requirement and the analysis model as a whole are validated against customer need to ensure that the right system is to be built.

Dictation

- | | | | | |
|--------------|------------------|-------------|----------------|-----------------|
| 1. engineers | 2. follows | 3. cases | 4. same | 5. projects |
| 6. working | 7. organizations | 8. group | 9. originally | 10. built |
| 11. overall | 12. requirements | 13. product | 14. department | 15. component |
| 16. contrary | 17. actually | 18. achieve | 19. purposes | 20. operational |

Part 2 Reading and Translating

Section A

- | | | | | | |
|-----|------|------|------|------|------|
| I. | 1. T | 2. F | 3. T | 4. T | 5. T |
| II. | 1. B | 2. D | 3. C | | |

III. 形成软件需求说明困难的主要原因是来自需求参与的三方——客户、最终用户和软件开发人员。

Section B

- I. 1. F 2. F 3. T 4. T 5. F
II. 1. D 2. B 3. C

- III. 1. 移动互联网 (mobile Web) 指的是使用诸如智能手机、功能手机等手持移动设备, 通过连接移动网络或其他无线网络来接入万维网, 如基于浏览器的互联网服务。
2. 移动带宽调制解调器可将智能手机“绑定”在一个或多个计算机或最终用户设备上, 通过蜂窝电话服务提供商所能提供的协议, 从而提供对互联网的连接。

Unit 3

Part 1 Listening & Speaking

Listening Comprehension

1. B 2. D 3. C

原文如下:

Software Project Planning

There are many different planning philosophies. Some people are “minimalists”, arguing that change often obviates the need for a detailed plan. Others are “traditionalists”, arguing that the plan provides an effective road map, and the more detail it has, the less likely the team will become lost. Still others are “agilists”, arguing that a quick “planning game” may be necessary, but the road map will emerge as “real work” on the software begins.

What to do? On many projects, over-planning is time consuming and fruitless (too many things change), but under-planning is a recipe for chaos. Like most things in life, planning should be conducted in moderation, enough to provide useful guidance for the team — no more, no less.

In an excellent paper on software process and projects, Barry Boehm states: “You need an organizing principle that scales down to provide simple plans for simple projects.” Boehm suggests an approach that addresses project objectives, milestones and schedules, responsibilities, management and technical approaches, and required resources. He calls it the W5HH principle, after a series of questions that lead to a definition of key project characteristic and the resultant project plan: Why is the system being developed? What will be done? When will it be accomplished? Those above are the most indispensable. In addition, who is responsible for a function? Where are their organizations located? How will the job be done technically and managerially? And how much of each resource is needed? The answers to Boehm’s W5HH questions are important regardless of the size or complexity of a software project.

Dictation

- | | | | | |
|---------------|----------------|----------------|----------------|-------------|
| 1. activities | 2. pre-defined | 3. finite | 4. process | 5. schedule |
| 6. classic | 7. control | 8. interacting | 9. over-budget | 10. quality |
| 11. within | 12. lifecycle | 13. saying | 14. baby | 15. visible |
| 16. choose | 17. implied | 18. inputs | 19. pick | 20. meeting |

Part 2 Reading and Translating

Section A

- I. 1. T 2. F 3. F 4. T 5. T
II. 1. B 2. C 3. D

- III. 1. 为了即使在风险存在的情况下能够实现项目目标, 需要适当的风险管理。
2. 通过适当的监控, 这些情况能够被识别, 从而可以相应地改变计划。

Section B

- I. 1. F 2. T 3. F 4. T 5. F
II. 1. D 2. B 3. C

- III. 然而, 为了从这些数据中获取商业价值, 企业需要能够对自己传统企业数据以及非传统数据进行管理和分析的新技术和新工具。

Unit 4

Part 1 Listening & Speaking

Listening Comprehension

1. C 2. B 3. D

原文如下:

Project Team

There are many reasons that software projects get into trouble. Many projects in which the scale of efforts is large need to be developed by project teams ranging in size from two to several hundred people. That is leading to complexity, confusion, and significant difficulties in coordinating team members. Putting together a group that works effectively is therefore a critical management task. A good group has a team spirit so that the people involved are motivated by the success of the group as well as by their own personal goals.

Ian Sommerville describes this spirit as “cohesiveness” in his book Software Engineering. In a cohesive group, members think of the group as more important than the individual in it.

Members of a well-led, cohesive group are loyal to the group. They identify with group goals and with other group members. They attempt to protect the group, as an entity, from outside interference. This makes the group robust and able to cope with problems and unexpected situations. The group can cope with change by providing mutual support and help. The advantages of a cohesive group include a group quality standard that is established by consensus and observed by self, group members working closely together, learning from each other, getting to know each other's work, practicing egoless programming, and so on.

In another book, *Peopleware*, DeMarco and Lister call that spirit as "jell". They discuss this issue as follows. A jelled team is a group of people so strongly knit that the whole is greater than the sum of the parts. Once a team begins to jell, the probability of success goes way up. The team can become unstoppable, a juggernaut or success... they do not need to be managed in the traditional way, and they certainly do not need to be motivated. They have got momentum.

To strengthen the cohesiveness and jell of a group, some effective methods must be established for coordinating the members to come to understand the motivations, strengths and weaknesses of other people in the group. Good communication between members of a software development group is one of the most essential ways. The group members must exchange information on the status of their work, the design decisions that have been made and changes to previous decisions that are necessary. To accomplish this, mechanisms for formal and informal communication among team members must be established. Formal communication is accomplished through "writing, structured meetings, and other relatively non-interactive and impersonal communication channels". Informal communication is more personal. Members of a software team share ideas on an ad hoc basis, ask for help as problems arise, and interact with one another on a daily basis.

Dictation

- | | | | | |
|---------------|------------------|-------------------|-------------------|----------------|
| 1. fast-paced | 2. ever-changing | 3. evolve | 4. emerge | 5. enough |
| 6. business | 7. Initially | 8. lightweight | 9. people-centric | 10. Alliance |
| 11. statement | 12. interactions | 13. documentation | 14. customer | 15. responding |
| 16. prior | 17. Extreme | 18. Adaptive | 19. Feature | 20. Dynamic |

Part 2 Reading and Translating

Section A

- | | | | | | |
|-----|------|------|------|------|------|
| I. | 1. T | 2. T | 3. F | 4. F | 5. T |
| II. | 1. C | 2. A | 3. C | | |

- III. 1. 详细进度安排只有在实际的人员分配已经完成之后才进行, 因为任务分配需要团队成员能力的相关信息。
2. 小组目标由全体成员的一致同意来设定, 并且每位成员的提议可用于重要决定。

Section B

- I. 1. F 2. T 3. T 4. T 5. F
II. 1. C 2. B 3. A

III. 互联网技术广泛应用在台式计算机、笔记本电脑、平板电脑、智能手机上，在民用电子、电器、汽车、医疗设备、公共事业系统、各种类型的机械装置，甚至衣服上——几乎所有可以装备传感器的东西，来收集信息、连接互联网，并且用数据分析软件进行数据分析。

Unit 5

Part 1 Listening & Speaking

Listening Comprehension

1. D 2. C 3. B

原文如下：

Software Design

Design engineering commences as the first iteration of requirements engineering comes to a conclusion. The intent of software design is to apply a set of principles, concepts, and practices that lead to the development of a high-quality system or product. The goal of design is to create a model of software that will implement all customer requirements correctly and bring delight to those who use it. Design engineers must sift through many design alternatives and converge on a solution that best suits the needs of project stakeholders.

The design process moves from a “big picture” view of software to a more narrow view that defines the detail required to implement a system. The process begins by focusing on architecture. Subsystems are defined; communication mechanisms among subsystems are established; components are identified; and a detailed description of each component is developed. In addition, external, internal, and user interfaces are designed.

Design concepts have evolved over the first half-century of software engineering work. They describe attributes of computer software that should be present regardless of the software engineering process that is chosen, the design methods that are applied, or the programming languages that are used.

The design model encompasses four different elements. As each of these elements is developed, a more complete view of the design evolves. The architectural element uses information derived from the application domain, the analysis model, and available catalogs for patterns and styles to derive a complete structural representation of the software, its subsystems and components. Interface design elements model external and internal interfaces and the user interface. Component-level elements define each of the modules (components) that populate the

architecture. Finally, deployment-level design elements allocate the architecture, its components, and the interfaces to the physical configuration that will house the software.

Dictation

- | | | | | |
|-------------------|---------------|----------------|----------------|-------------------|
| 1. Human-Computer | 2. experience | 3. rely | 4. usable | 5. acceptance |
| 6. marketability | 7. process | 8. focus | 9. specialists | 10. relationship |
| 11. control | 12. blame | 13. Contrarily | 14. accomplish | 15. well-designed |
| 16. behaves | 17. place | 18. load | 19. consistent | 20. golden rules |

Part 2 Reading and Translating

Section A

- I. 1. T 2. T 3. F 4. T 5. T
- II. 1. D 2. D 3. C

III. 以数据结构为中心的设计（如 Jackson 方法、Warnier-Orr 方法）从程序操作的数据结构着手，而不是从其实现的功能着手。

Section B

- I. 1. F 2. F 3. T 4. T 5. F
- II. 1. D 2. A 3. D

III. 1. 社交网络的另一个好处就是多样性，因为互联网允许世界各地的人们都可以接入社交站点。

2. 最为普遍的风险往往和互联网上的“掠夺者”或虚假身份有关。

Unit 6

Part 1 Listening & Speaking

Listening Comprehension

1. A 2. D 3. B

原文如下：

Writing the Code

So far, the skills we have learned helped us to understand the customers “and users” problem and to devise a high-level solution for it. Now, we must focus on implementing the solution as software. That is, we must write the programs that implement the design. This task can be daunting, for several reasons.

First, the designers may not have addressed all of the characteristics of the platform and programming environment; structures and relationships that are easy to describe with charts and

tables are not always straightforward to write as code.

Second, we must write our code in a way that is understandable not only to us when we revisit for testing, but also to others as the system evolves over time. In the real world, most software is developed by teams, and a variety of jobs are required to generate a quality product. Even when writing the code itself, many people are usually involved, and a great deal of cooperation and coordination is required. Thus, it is very important for others to be able to understand not only what you have written, but also why you have written it and how it fits in with their work. To be a truly excellent programmer, you must know that programming is communicating with other programmers first and communicating with the computer second.

Third, we must take advantage of the characteristics of the design's organization, the data's structure, and the programming language's constructs while still creating code that is easily reusable. The most critical standard is the need for a direct correspondence between the program design components and the program code components. The entire design process is of little value if the design's modularity is not carried forward into the code. Design characteristics, such as low coupling, high cohesion, and well-defined interfaces, should also be program characteristics, so that the algorithms, functions, interfaces, and data structures can be traced easily from design to code and back again. In case of requirements changing, they are made first to the high-level design and then are traced through lower design levels to the code that must be modified.

Although much of coding is an individual endeavor, all of coding must be done with your team in mind. Your use of information hiding allows you to reveal only the essential information about your components so that your colleagues can invoke them or reuse them. Your use of standards enhances communication among team members. And your use of common design techniques and strategies makes your code easier to test, maintain, and reuse. No matter which programming language to be used, and no matter how the code to be worked out, you should keep some of the software engineering practices above in mind as you write your code.

Dictation

- | | | | | |
|----------------|---------------|----------------|--------------|--------------|
| 1. effects | 2. causes | 3. minority | 4. greatest | 5. quality |
| 6. named | 7. economist | 8. population | 9. Similarly | 10. follow |
| 11. contribute | 12. execution | 13. experience | 14. program | 15. loops |
| 16. doubled | 17. hot spots | 18. resources | 19. percent | 20. multiply |

Part 2 Reading and Translating

Section A

- | | | | | | |
|-----|------|------|------|------|------|
| I. | 1. T | 2. F | 3. T | 4. T | 5. F |
| II. | 1. D | 2. D | 3. C | | |

III. 伪代码是使用标准人类语言表述来描述逻辑和处理流程的设计程序的一种方法。

Section B

- I. 1. F 2. F 3. F 4. T 5. F
II. 1. B 2. D 3. A

- III. 1. 基于位置的服务 (LBS) 是一种计算机程序级别的服务, 用来将时间和位置数据的特定控制作为控制特性包含在计算机程序中。
2. LBS 还包括移动商务, 如基于用户的当前位置来发布广告和优惠券。

Unit 7

Part 1 Listening & Speaking

Listening Comprehension

1. B 2. A 3. C

原文如下:

Software Testing

Testing plays a critical role in quality assurance for software. Due to the limitations of the verification methods for the previous phases, design and requirement faults also appear in the code. Testing is used to detect these errors, in addition to the errors introduced during the coding phase.

Testing is a dynamic method for verification and validation, where the system to be tested is executed and the behavior of the system is observed. Due to this, testing observes the failures of the system, from which the presence of faults can be deduced. However, separate activities have to be performed to identify the faults and then remove them.

There are two approaches to testing: black-box and white-box. In black-box testing, the internal logic of the system under testing is not considered and the test cases are decided from the specifications or the requirements. It is often called functional testing. Equivalence class partitioning, boundary value analysis and cause-effect graphing are examples of methods for selecting test cases for black-box testing. State-based testing is another approach in which the system is modeled as a state machine and then this model is used to select test cases using some transition or path based coverage criteria. State-based testing can also be viewed as grey-box testing in that it often requires more information than just the requirements.

In white-box testing, the test cases are decided entirely on the internal logic of the program or module being tested. The external specifications are not considered. Often a criterion is specified, but the procedure for selecting test cases is left to the tester. Mutation testing is an approach for white-box testing that creates mutants of the original program by changing the original program. The testing criterion is to kill all the mutants by having the mutant generate a different output from the original program.

As the goal of testing is to detect any errors in the programs, different levels of testing are often used. Unit testing is used to test a module or a small collection of modules and the focus is on detecting coding errors in modules. During integration testing, modules are combined into subsystems, which are then tested. The goal here is to test the system design. In system testing and acceptance testing, the entire system is tested. The goal here is to test the system against the requirements, and to test the requirements themselves. White-box testing can be used for unit testing, while at higher levels mostly black-box testing is used.

Dictation

- | | | | | |
|----------------|----------------|----------------|----------------|------------|
| 1. smoke | 2. assembly | 3. validating | 4. checked | 5. defects |
| 6. confirm | 7. expected | 8. destabilize | 9. integration | 10. build |
| 11. exhaustive | 12. exposing | 13. benefits | 14. critical | 15. risk |
| 16. conducting | 17. uncovering | 18. correction | 19. assessment | 20. a bit |

Part 2 Reading and Translating

Section A

- I. 1. T 2. F 3. T 4. T 5. T
- II. 1. B 2. B 3. D

- III. 1. 测试是一项用于通过识别缺陷和问题来评估和改进产品质量而进行的活动。
2. 现代系统化的集成策略更倾向于是体系结构驱动的, 指基于已识别的功能性部分来集成软件组件或子系统。

Section B

- I. 1. F 2. F 3. T 4. T 5. F
- II. 1. C 2. B 3. D

- III. 1. 不难想象, 基于像素的应用(如计算机视觉、视频、图像处理)非常适合于 GPGPU 技术来处理, 因为这个原因, 很多这些领域的商业软件包都包含了 GPGPU 加速。
2. 为了达到这样的提速, 算法必须根据 GPU 的结构来编码, 而针对 GPU 的编程与针对传统的 CPU 有很大的不同。

Unit 8

Part 1 Listening & Speaking

Listening Comprehension

1. C 2. A 3. C

原文如下:

Software Delivery

The delivery of a software increment represents an important milestone for any software project. But before your cheer for that, something important must be completed. First of all, you must ensure that your delivery should have been tested and fixed without bugs. Second, you should deliver more than the software itself. That is to say, besides executable software, support data files, support documents, and other relevant information must be assembled together. An appropriate instructional materials must be provided to end-user such as appropriate training aids, trouble-shooting guidelines, and a “what’s-different-about-this-software-increment” description. Furthermore, a support regime must be established before the software is delivered. An end-user expects responsiveness and accurate information when a question or problem arises. If support is ad hoc, or worse, nonexistent, the customer will become dissatisfied immediately. Support should be planned, support material should be prepared, and appropriate record keeping mechanisms should be established so that your software team can conduct a categorical assessment of the kinds of support requested.

The delivered software provides benefit for the end-user, but it also provides useful feedback for the software team. As the increment is put into use, the end-users should be encouraged to comment on features and functions, ease of use, reliability, and any other characteristics that are appropriate. Feedback should be collected and recorded by the software team and used to make immediate modifications to the delivered increment, define changes to be incorporated into the next planned increment, make necessary design modifications to accommodate changes, and revise the plan for the next increment to reflect the changes.

Dictation

- | | | | | |
|---------------|-------------|----------------|-----------------|--------------|
| 1. attributed | 2. observed | 3. defects | 4. errors | 5. entering |
| 6. indication | 7. external | 8. internal | 9. relationship | 10. connects |
| 11. locating | 12. device | 13. commercial | 14. application | 15. portion |
| 16. fail | 17. quit | 18. divide | 19. trail | 20. fresh |

Part 2 Reading and Translating

Section A

- | | | | | | |
|-----|------|------|------|------|------|
| I. | 1. T | 2. F | 3. T | 4. T | 5. T |
| II. | 1. C | 2. A | 3. C | | |

- III. 1. 设计组件约束于需求规格说明, 代码组件被对照和评审以遵从于设计, 测试基于找出功能和约束是否依据需求和设计工作。
2. 分析员或程序员确定代码中哪些部分受到了影响, 其对于设计的影响以及可能的资源 (包括时间和工作量), 以做出必要的更改。

Section B

- I. 1. F 2. F 3. T 4. F 5. T
II. 1. D 2. C 3. D

- III. 1. 这对货币的流通起到了刺激的作用,但同时也吸引了很多正在寻找不义之财的掘金者和投机商人,以及那些自认为可以榨取痴迷者的骗子们。
2. 这些服务器在基于互联网的网络上进行通信,并通过将其加入总账的方式确认交易。总账使用 P2P 文件共享技术定期地进行更新和存档。

Unit 9

Part 1 Listening & Speaking

Listening Comprehension

1. B 2. A 3. C

原文如下:

Expansion Company

As a part of job invitation advertisement, here is an overview on Expansion, a company providing software products and services in IT industry:

Expansion is one of the top software products, services and solutions providers in China. It was established in 1990, and now is a leading IT company in China, but also has a global presence. It headquarters in the heart of China's Silicon Valley, Zhong Guancun, Beijing. Expansion helps to meet the outsourcing needs of companies worldwide, and its goal is to become a global leading IT outsourcing service company by providing quality services to global customers.

In the early years of Expansion, testing service was its major business. With the development of the company, there are more and more various products and services to provide today. Among them, Software Development and Application Maintenance Service have become the key business. In addition, BPO, shorted for Business Process Outsourcing, is a kind of rising service which Expansion begins trying.

In the past decade, Expansion has a brilliant track record as an IT partner for Fortune 500 companies, including Microsoft, HP, IBM, and so on. Japan is the main customer among its overseas market at present. Meanwhile, Europe and U.S. will become another two huge markets shortly by establishing its offices there.

Expansion has an experienced management team and a rich pool of well educated, well trained and highly experienced engineers and project managers. Further, its staffs are expected to grow over 900 members in 2014. Through the endeavor from all of them, Expansion has been keeping

margin increase at the rate of over 10% every year since foundation.

In the future, Expansion will attract new clients continually to develop global presence, advance strong management team and human resources by investing in employees' trainings to develop their experience, and become more profitable by high efficiency and top quality in work. By all of those efforts, Expansion will be very hopeful to realize its strategic goal.

Dictation

- | | | | | |
|------------------|-------------------|---------------|--------------|-----------------|
| 1. change | 2. effectively | 3. control | 4. likely | 5. establishing |
| 6. versions | 7. reporting | 8. eventually | 9. detect | 10. integration |
| 11. requirements | 12. documentation | 13. idea | 14. kept | 15. check out |
| 16. copies | 17. back | 18. revisions | 19. features | 20. versioning |

Part 2 Reading and Translating

Section A

- I. 1. T 2. T 3. F 4. T 5. F
- II. 1. B 2. D 3. A

- III. 1. 计算机程序设计要求专心致志和对于与一个程序设计项目有关的无数细节的好记忆力。
2. 安全专家必须拥有广泛的计算机知识以及能够应用于对所出现的任何危机的快速解决方案的通信协议的知识。

Section B

- I. 1. F 2. T 3. T 4. T 5. T
- II. 1. A 2. D 3. B

- III. 这是一种在能源利用、废物处理、定制、产品可用性、艺术、医药、建筑、科学技术以及生产等方面的颠覆性技术。

Unit 10

Part 1 Listening & Speaking

Listening Comprehension

1. A 2. B 3. A

原文如下:

The Organizational Structure of a Company

It is important that every new member gets to know the organizational structure of his company and realizes the relationships of membership as soon as possible. The brief description below about the company organization will facilitate the understanding of the company that a new employee works for.

In a company, the managers and subordinates make up an organization. There are some main chiefs taking the lead. The first one is CEO, which is shortened form of Chief Executive Officer. He is the executive who is responsible for a company's overall operations, usually is the President or/and the Chairman of the Board of a company. In addition, Chief Financial Officer, he is senior manager who is responsible for overseeing the financial activities of an entire company, including financial planning and cash flow, CFO is for short. Another senior manager is CTO, whose full name is Chief Technology Officer. He is in charge of technology related departments in the company such as department of IT Support and department of Information System. Besides, COO is the acronym of Chief Operating Officer, who is senior manager whose responsibility is to manage the day-to-day operations of the company and then report this to the CEO. They form the senior leadership of the company.

The organization mentioned above is classed in terms of rank. So there is a hierarchical structure in a company: the CEO is the head of the company; the CTO, CFO and COO report directly to the CEO; the Director of Marketing and the Manager of Customer Services are under the supervision of the COO. As this company grows, its organization may become much larger and the real structure is likely to be much more complex. That is to say, every position in this structure can be expandable, maybe the CTO and the CFO may each have teams working under them. In some companies, there are CBDMO, the Chief Business Development and Marketing Officer, and CSO, the Chief Strategy Officer besides.

Dictation

- | | | | | |
|---------------|----------------|------------------|------------------|-----------------------|
| 1. bundled | 2. withholding | 3. business | 4. secret | 5. distributed |
| 6. harnesses | 7. lock-in | 8. protocols | 9. characterized | 10. birth |
| 11. session | 12. reaction | 13. announcement | 14. innovative | 15. operating systems |
| 16. generated | 17. challenge | 18. exception | 19. procedures | 20. mechanisms |

Part 2 Reading and Translating

Section A

- | | | | | | |
|-----|------|------|------|------|------|
| I. | 1. T | 2. F | 3. T | 4. F | 5. T |
| II. | 1. A | 2. C | 3. D | | |

III. 1. 人力资源（或人事）部门寻找和雇用人员，并管理病假和退休事宜。

2. 信息系统部门管理该组织基于计算机的系统, 并计划和购买新系统。

Section B

I. 1. F 2. F 3. T 4. T 5. F

II. 1. D 2. B 3. A

III. 方向变得更加直观 (尽管现在看来谷歌眼镜中没有加入 GPS, 所以需要与手机一起使用); 可以对谈话进行实时翻译和记录; 也可以查看和回复信息, 所有这些都是在在线进行的。

Bibliography (参考文献)

1. Frederick P. Brooks., Jr. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition [M]. 2d ed. Reading, MA: Addison-Wesley, 1995.
2. Roger S Pressman. Software Engineering: A Practitioner's Approach [M]. 6th edition. New York, NY: McGraw-Hill, 2005.
3. Kent Beck. Extreme Programming Explained: Embrace Change [M]. Reading, MA: Addison-Wesley, 2000.
4. Ian Sommerville. Software Engineering [M]. 8th edition. Addison Wesley, 2005.
5. Steven C McConnell. Code Complete [M]. 2nd edition. Microsoft Press, 2004.
6. Andrew Hunt, David Thomas. The Pragmatic Programmer [M]. Addison Wesley, 2003.
7. David, J Agans. Debugging: The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems [M]. Microsoft Press, 2002.
8. Steven C McConnell. Rapid Development: Taming Wild Software Schedules [M]. Microsoft Press, 1996.
9. Brian K Williams, Stacey C Sawyer. Using Information Technology: A Practical Introduction to Computers & Communications [M]. 7th ed. New York, NY: McGraw-Hill, 2007.
10. June Jamrich Parsons, Dan Oja. New Perspectives on Computer Concepts, 2014, Comprehensive [M]. Cengage Learning, 2014.
11. Timothy J O'Leary, Linda I O'Leary. Computing Essentials 2007 [M]. New York, NY: McGraw-Hill, 2007.
12. Kenneth C Laudon, Carol Guercio Traver. E-commerce 2013: Business. Technology. Society. 9th edition [M]. Pearson Education, 2013.
13. Misty E Vermaat. Discovering Computers 2014 [M]. 1st ed. Cengage Learning, 2014.