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

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。

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

产品编号：054675-01

FOREWORD

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时数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%。

Contents

目　录

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 proyress 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
['sɔfəmɔ:] n. 大学二年级学生

rough
[rʌf] adj. 初步的，粗略的

specification
[ˌspesifi'keiʃən] n. 说明书，规范

reminder
[ri'maində] 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:

1. Which is correct about the development of software according to the article?
   (A) It emerged with software engineering at the same time.
   (B) For a half-century development, it has almost solved problems of high-quality, on-time and within-budget.
   (C) It was just a specialized problem solving and information analysis tool in its early years of development.
   (D) The laws which software evolves according to have changed absolutely during its development.

2. Which point does not belong to the characteristics of software according to the article?
   (A) Easy to change the requirements
   (B) Easy to adapt the requirement changes
   (C) Difficult to measure the progress and process of creating
   (D) Difficult to test the correctness exhaustively

3. Where was the phrase "software engineering" first used in 1968?
   (A) In a conference
   (B) In a thesis
   (C) In a journal

(D) In a magazine

Words

demonstrate
['demənstreit] v. 证明，论证

concern
[kən'sə:n] v. 担心，忧虑

practice
['præktis] n. 实践，实行

assimilate
[ə'simileit] v. 吸收

intangible
[in'tændʒəbl] adj. 无形的

exhaustively
[ig'zɔ:stivli] adv. 详尽地，彻底地

address
[ə'dres] v. 处理，满足

law
[lɔ:] n. 规则，法则

framework
['freimwə:k] n. 构架，体系结构

prototype
['prəutətaip] n. 原型

discipline
['disiplin] 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'vɔkətiv] 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