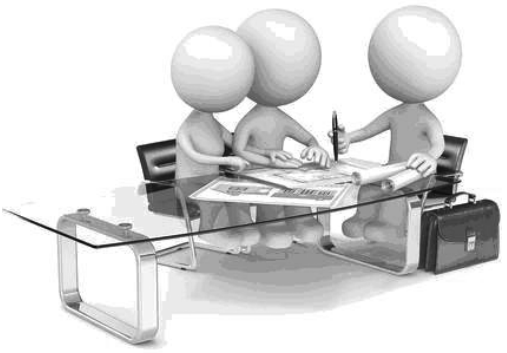
Unit Three Software Engineering

Section A Software Engineering Methodologies



Software development is an engineering process. The goal of researchers in software engineering is to find principles that guide the software development process and lead to efficient, reliable software products.

methodologies
[ˌmeθədɒˈlɒdʒiz]
n. 方法论

Ⅰ. Waterfall Model

The waterfall model(see Figure 3A-1) is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards through the phases of requirements analysis, design, implementation, testing, and maintenance.

waterfall model
瀑布模型

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are **prohibitively** costly. The first known presentation describing use of similar phases in software engineering was held by Herbert D.Benington at **Symposium** on advanced programming methods for digital computers[①] on 29 June 1956.

prohibitively
[prəˈhɪbətɪvli]
adv. 过高地

symposium
[sɪmˈpəʊziəm]
n. 研讨会

① 赫伯特·D.贝宁顿在数字计算机高级编程方法研讨会上第一次在软件工程中提出了类似的阶段。

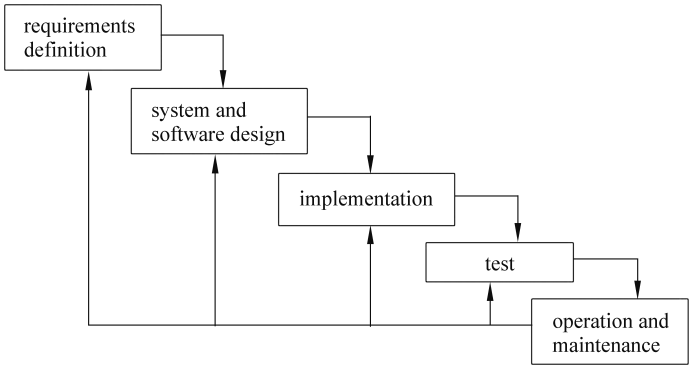


Figure 3A-1 Waterfall Model

Ⅱ. Incremental Model

In recent years, software engineering techniques have changed to reflect the **contradiction** between the highly structured environment **dictated** by the waterfall model and the "**free-wheeling**", **trial-and-error** process that is often vital to creative problem solving. This is illustrated by the emergence of the **incremental model**(see Figure 3A-2) for software development. Following this model, the desired software system is constructed in increments—the first being a simplified version of the final product with limited functionality. Once this version has been tested and perhaps **evaluated** by the future user, more features are added and tested in an incremental manner until the system is complete.

contradiction
n. 矛盾
dictate
v. 控制;命令
free-wheeling
单向转动
trial-and-error
试错(法);不断摸索
incremental model
增量模型
evaluate
v. 评估

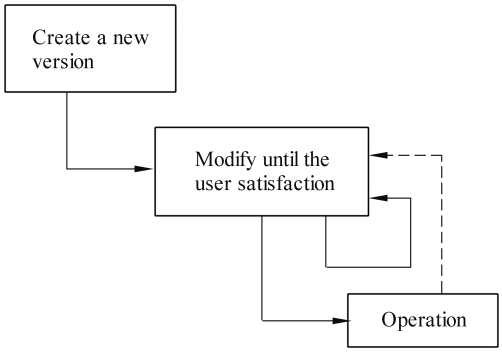


Figure 3A-2 Incremental Model

Ⅲ. Iterative Model

Another model that represents the shift away from strict adherence to the waterfall model is the **iterative model** (see Figure 3A-3), which is similar to, and in fact sometimes equated with, the incremental model, although the two are distinct. Whereas the incremental model carries the notion of extending each **preliminary** version of a product into a larger version, the iterative model **encompasses** the concept of **refining** each version. In reality, the incremental model involves an underlying iterative process, and the iterative model may incrementally add features.

iterative
/ˈɪtərətɪv/
adj. 迭代的
preliminary
/prɪˈlɪmɪneri/
adj. 初步的
encompass
/ɪnˈkʌmpəs/
vt. 围绕
refine
/rɪˈfaɪn/
vt. 提炼；精炼

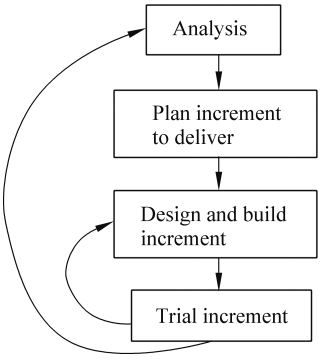


Figure 3A-3 Iterative Model

A significant example of iterative techniques is the **rational unified process** (**RUP**) (see Figure 3A-4) that was created by the Rational[①] Software Corporation, which is now a division of IBM. Generally, RUP is a development plan, which specifies the general process of developing a software product. Precisely, definition that states, which activities are to be performed, by which person, acting in which role; in which order the activities will be performed, and which products will be developed and how to evaluate them.

rational unified process
统一软件过程

Incremental and iterative models sometimes make use of the trend in software development toward **prototyping** in which incomplete versions of the proposed system, called prototypes, are

prototyping
/ˈprəʊtətaɪpɪŋ/
n. 原型设计制作

① Rational：提供基于业界开放标准的工具、最佳方案和服务，用于开发商业应用和构建软件产品及系统，包括移动电话和医疗系统等设备使用的嵌入式软件。

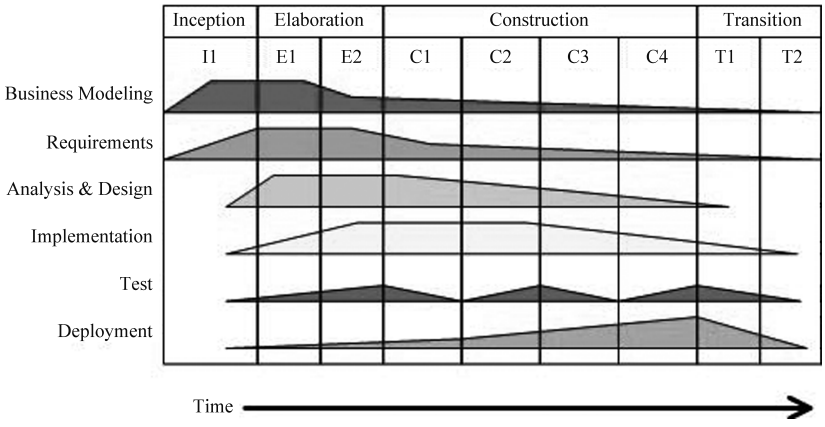


Figure 3A-4 The Unified Process

built and evaluated.

A less formal **incarnation** of incremental and iterative ideas that has been used for years by computer enthusiasts is known as **open-source development.** This is the means by which much of today's free software is produced. Perhaps the most prominent example is the Linux operating system whose open-source development was originally led by Linus Torvalds[①].

incarnation
[ˌɪnkɑːˈneɪʃn]
n. 化身
open-source development
开源软件的开发

Ⅳ. Agile Methods

Perhaps the most **pronounced** shift from the waterfall model is represented by the agile methods, each of which proposes early and quick implementation on an incremental basis, **responsiveness** to changing requirements, and a reduced emphasis on **rigorous** requirements analysis and design. One example of an agile method is **extreme programming(XP).**

XP was created by Kent Beck[②] during his work on the Chrysler Comprehensive Compensation System payroll project in the 1990's. In 1999, Beck defined XP as a lightweight methodology for the small-to-medium sized teams developing software in the face of vague or rapidly changing requirements. In terms of this definition,

agile methods
敏捷方法
pronounced
[prəˈnaʊnst]
adj. 明显的
responsiveness
[rɪˈspɒnsɪvnəs]
n. 敏感性
rigorous
[ˈrɪgərəs]
adj. 严厉的;严密的
extreme programming(XP)
极限编程

① Linus Torvalds: 林纳斯·托瓦尔兹(1969—),芬兰人,著名的计算机程序员、黑客。Linux 内核的发明人及该计划的合作者。

② Kent Beck: 肯特·贝克(1961—),美国人,软件开发大师,是最早研究软件开发模式和重构的人之一,是敏捷方法的开创者之一,对当今世界的软件开发影响深远。

XP could be described in four ways. Firstly, it is lightweight. In XP you only do what you need to do to create value for the customer. Secondly, it is a methodology, which based on addressing obstacle in software development. It does not address some specific issues such as financial justification of projects or sales. Thirdly, it is implemented by a small or medium size, which makes the project team much more flexible. In this kind of team, everyone is elite and has his clear responsibility. Last but not least, XP adapts to rapidly changing requirements. In the modem business world requirements need to change to adapt to rapid shifts, so that the vital role that XP plays on the operating system cannot be over-emphasized.

Figure 3A-5 The Logo of XP

The contrasts depicted by comparing the waterfall model and XP reveal the **breadth** of methodologies that are being applied to the software development process in the hopes of finding better ways to construct reliable software in an efficient manner. Research in the field is an ongoing process. Progress is being made, but much work remains to be done.

breadth

n. 宽度;广泛

Exercises

Ⅰ. Fill in the blanks with the information given in the text.

1. The goal of researchers in _____ is to find principles that guide the software development process and lead to efficient, reliable software products.

2. Early approaches to software engineering insisted on performing requirements analysis, _____, implementation, _____ and maintenance in a strictly sequential manner.

3. Following incremental model, the desired software system is constructed in _____— the first being a simplified version of the final product with limited functionality.

4. A significant example of iterative techniques is the _____, which specifies the general process of developing a software product.

5. The most prominent example is the _____ operating system whose open-source development was originally led by Linus Torvalds.

6. Following the _____ model, software is developed by a team of less than a dozen individuals working in a communal work space where they freely share ideas and assist each other in the development project.

Ⅱ. Translate the following terms or phrases from English into Chinese.

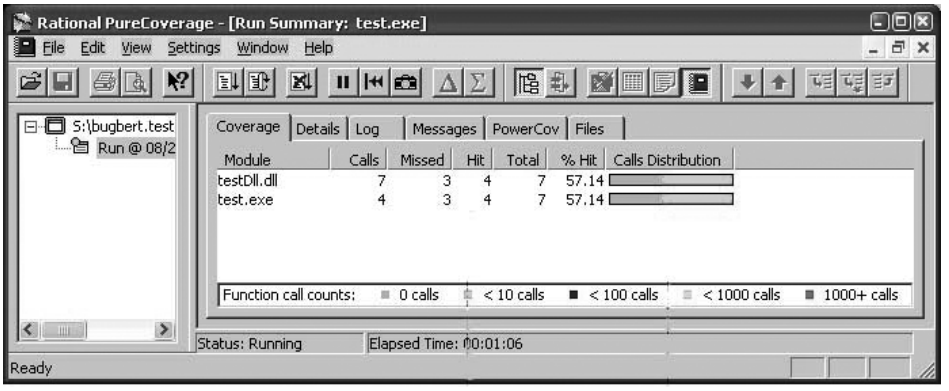
software engineering	requirements analysis
testing	maintenance
software engineer	requirements specification
waterfall model	incremental model
iterative model	rational unified process(RUP)
software life cycle	open-source
agile method	extreme programming(XP)

Ⅲ. Translate the following passage from English into Chinese.

Software Ownership

Legal efforts to provide such ownership fall under the category of intellectual property law, much of which is based on the well-established principles of copyright and patent law. Indeed, the purpose of a copyright or patent is to allow the developer of a "product" to release that product to intended parties while protecting his or her ownership rights. As such, the developer of a product (whether an individual or a corporation) will assert his or her ownership by including a copyright statement in all produced works; including requirement specifications, design documents, source code, test plans, and in some visible place within the final product. A copyright notice clearly identifies ownership, the personnel authorized to use the work, and other restrictions. Furthermore, the rights of the developer are formally expressed in legal terms in a software license.

Section B　Exploratory Testing



Software testing is a process where a software tester/team runs a program or a system to find **bugs** or defects, to maintain the correctness and reliability of a program. Software testing also validates and verifies the program to check if the business and technical requirements are met, and is working as expected.

bug
n.(软件)错误;漏洞

Ⅰ. What is Exploratory Testing?

exploratory testing
探索性测试

Exploratory testing—a term coined by Cem Kaner[①]—is a software testing method that implements learning, test design, and test execution at the same time simply because you explore while testing. It really doesn't follow a set of procedures, scripts or standards that is why it is mistakenly known as the "do it yourself" or careless activity, which is in fact, this approach describes an intellectual testing. During testing, the tester knows how the software really works, develop test design based on learning and experience, and the control to perform the entire testing stage. Because of the freedom that the exploratory testing offers, it solely depends on the tester's skill to find bugs, create **test cases**, and the responsibility to deliver the project optimizing the quality and value of the software.

test case
测试用例

① Cem Kaner: 卡尼尔,美国人,佛罗里达理工大学教授,在软件测试领域很出名。

Ⅱ. Situational Behavior of Exploratory Testing

Unlike **scripted testing** that follow procedures, the exploratory testing is extensive in such a way that it is a **trial and error** situation. Performing this approach requires keen observation of the behavior, thorough investigation, critical thinking/**tactics** in finding bugs, analyze the possible issues, and evaluate the entire software.

scripted testing
脚本测试
trial and error
反复试验
tactic
[ˈtæktɪk]
n. 战术

A situational example for this is playing an online game. At first, the user has to know what it is all about, the objective, and how to play the game. While playing, the user now identifies the environment and functionalities of the game. The user discovers then the encountered difficulties and foresees potential problems on each level. At the same time, the user thinks strategies/solutions and also executes them as well and be able to complete and achieve the objective of the game.

Therefore, relating the situational example to exploratory testing, the following are the keys to remember:

(a) Identify the project's purpose.

(b) Determine the requirements and functionalities of the software.

(c) Identify the limitation of the software.

(d) Test the functionalities thoroughly based on the requirements.

(e) Create test cases that verify the consistency of the software.

Ⅲ. Advantages

- Less preparation before the implementing the software testing because it doesn't require writing repeatable test procedures.
- Testing, designing and executing the software simultaneously saves time.
- Report many issues caused by incomplete or wrong documentation.

Ⅳ. Disadvantages

- Test designs created can't be reviewed prior to actual testing which might produce possible issues.
- Minor issues are poorly detected.
- Difficulty to perform the exact manner especially for new found bugs.

Ⅴ. Conclusion

Exploratory Testing is necessary to practice because it enhances the tester's mind and strategic skills through exploration and experience. Therefore, the more the tester knows the keys in using this method, the better the testing, and the quality of software project will be.

Exercises

Ⅰ. Fill in the blanks with the information given in the text.

1. Software testing is a process where a software tester/team runs a _____ or a system to find _____ or defects, to maintain the correctness and reliability of a program.

2. Exploratory testing is a software testing method that implements learning, test design, and test execution at the same time simply because you _____ while testing.

3. Exploratory testing solely depends on the tester's _____ to find bugs, create test cases.

4. Exploratory testing has less _____ before the implementing the software testing.

Ⅱ. Translate the following terms or phrases from English into Chinese.

software testing	exploratory testing
scripted testing	trial and error
bug	test case

Section C 计算机专业英语翻译

1. 翻译的要点和标准

翻译就是把一种语言表达的信息用另一种语言表达出来。这里主要介绍把计算机英文资料用中文加以表达的方法和技巧。翻译不仅要求对英文原文有透彻、正确的理解，而且要求用符合中文语言习惯的适当表达方式进行再表述；不仅需要计算机专业知识，而且要求掌握必要的文化与背景知识。

翻译有翻译的标准。在我国，关于翻译标准问题影响最大的，当推严复的“信、达、雅”。“信”就是要忠实于原文，准确译出原意和原味。“达”就是要达意，语言要适当，表达要能让人懂。“雅”就是要文采，译文要给人以赏心悦目的美感。就计算机专业英文翻译而言，主要是达到前两条标准。

翻译的方法可分为“直译”和“意译”两大类。对于技术型较强的资料，译者不熟悉、新的、前沿性的内容，应“直译”，力求准确，保证其与严谨性；对于广告性资料，译者十分了解、熟练掌握的内容，不妨“意译”，可以使译文更易读。在具体的翻译实践中，应尽量兼顾这两方面。

2. 英语科技文章在句子结构上的特点

除了掌握计算机专业知识，熟记大量的计算机专业英语词汇，还需要了解计算机专业英语句子在结构上的特点。归纳为以下“八多”：

- 陈述句多(主系表结构占较大比重)。
- 祈使句多(科技文章可操作性强，有时无须指明主语)。
- 被动句多(科技文章侧重叙事推理，强调客观准确)。
- 复合句多(并列关系、多种主从关系和非谓语动词构成的长句多)。
- 虚拟语气多(因科技文章常涉及各种条件)。
- 三种基本时态多(即一般现在时、一般过去时、现在完成时多)。
- as 引出的句式多(如 as shown in figure2，as stated above 等)。
- “It be＋形容词或分词＋that …”句型多。

3. 计算机专业英语的翻译技巧

翻译有一定的规律和方法可循。但是，翻译是一门需要大量实践的技能，只有通过自己的实践，并不断总结经验，才能提高翻译水平。下面讨论计算机专业英语翻译的一些常用方法，供大家在翻译实践中适当运用。

3.1 词义选择

每个英语单词都有一个或几个具体的含义，这就需要在翻译时选择一个恰当的词义。计算机英语是一种专业英语，其专业词汇大多是英语中已有的单词，只不过通过借用或引申原义等手段变成了计算机专业词汇。在具体的翻译中，可以根据上下文以及词的类别、

词性、搭配等方面来判断和选择词义。

例句 1：

The two main types of storage devices are disk drives and memory. 存储设备主要分两种类型：外存和内存。

在计算机专业英语中，storage 通常翻译成“存储”，memory 有“记忆、存储器、内存、回忆”等含义，但是根据上下文内容，此处介绍存储设备，那么 memory 应该翻译成“内存”，disk drives 本身有“磁盘驱动器”的意思，既然 memory 翻译成“内存”了，那么根据计算机知识翻译的习惯，此处的 disk drives 就翻译成“外存”。

例句 2：

I have to install with each new PC: Office Suite, including E-mail client...我给每台新计算机必须安装的应用程序有：Office 套件，包括电子邮件客户端……

In client/server computing, processing is distributed between two machines—the client machine and the server machine. 在客户机/服务器计算中，处理工作分布在两台机器上进行——客户机和服务器。

用作计算机专业词汇时，client 可以表示“客户程序”（软件领域），也可以表示“客户机”（硬件领域）。第一句话中的 client 显然指程序，第二句中的 client 与 server 相对，显然指硬件。

另外需要说明的是：有的计算机术语不加翻译可以直接借用到汉语中，并已经为 IT 业所惯用，如句中出现的 Office，还有 Java、IBM 等。

例句 3：

Because transistors use much less power and have a much longer life...因为晶体管的功率更小，寿命更长……

power 有“权力、力量、功率、动力、政权”等意思，根据上下文，此处介绍晶体管与电子管的区别，结合对当前计算机业的了解，生产的硬件大小有越来越小的趋势，而运算能力、使用寿命、功率等都有所提高。因此，此处的 power 翻译成“功率”更合理。

3.2 词性、句子成分、句子类型转换

英语和汉语之间有许多不同，其中对翻译影响较大的就是英语多用形合，汉语多用意合。所谓形合，就是在形式上加以配合，使用复句或从句，用一种语法形式来表达它们之间的句法关系，句中逻辑关系很严密，通常使用关联词语来实现。所谓意合，就是在意思上加以配合，通过上下文的逻辑关系来约束各个分句之间的关系。

英译汉时，如果拘泥于原文单词或句子的语法特征，势必产生拗口的句子甚至难以理解的句子。因此，在忠实于原文的前提下，要敢于突破语法框框，对词性、句子成分、句子类型等进行变通转换，以使译文自然流畅，符合汉语习惯。

例句 4：

The new electronic computer is chiefly characterized by its simplicity of structure. 这种新型计算机的主要特点是结构简单。

本句涉及副词转换为形容词，即副词 chiefly 译文中变成了“主要的”。当英语动词 characterize 在翻译时转换为汉语名词时，原来相关的状语副词往往可以译为形容词。

例句 5：

A dominant factor in the growth of MS in question throughout the years has been its success in maintaining technical superiority in product development. 微软公司这些年来发展壮大的主要因素是其一直成功地保持了在产品开发方面的技术优势。

本句涉及名词转换为副词，即名词 success 在译文中变成了副词“成功地”。

例句 6：

You should use your E-mail account with the expectation that some of your mail will be read from time to time. 使用电子邮件账号时，你应该想到：你的一些邮件会被人不时地阅读。

本句涉及名词转换为动词，即名词 expectation 在译文中变成了动词“想到”。此外，原文的句子成分在译文中也发生了变化：主句的谓语动词 use 及其宾语 your E-mail account 转为时间状语；主句作状语的介词短语 with the expectation 转为主句的谓语动词；同位语从句转为宾语从句。这句话如果按照原文的单词词性和句子成分翻译，会非常拗口：你应该带着你的一些邮件会被人不时阅读的料想使用你的电子邮件账号。

例句 7：

The properties of this computer should be made full use of. 应该充分利用这台计算机的性能。

本句的主语转换为宾语。

例 8：

The first inventor of this precision instrument seems to have been Charles Babbage in 1847—the same man who is regarded as the father of the computer. 计算机之父查尔斯·巴贝奇在 1847 年首先发明了这种精密仪器。

本句是表语转换为主语。

3.3 增词与减词

在英译汉时，译文增添或者省略一些词，是为了更好地表达原文的意思，便于读者理解原文。但是，补充只能在原文的基础上和作者的意思内进行，不能离题发挥。省略的词必须是译文中确实不需要表达的，不能损及原文的意思。

例句 9：

Computers have opened up a new era in manufacturing through the techniques of automation, and they have enhanced modern communication systems. 通过自动化技术，计算机开辟了制造业的新纪元，也增强了现代通信系统的性能。

本句通过增词将原文的意思表达完整。

例句 10：

It would be easy to assume that computers will all be able to work together once they have broadband connection. 人们很容易认为：计算机一旦有了宽带连接，就能全部具备协同工作的能力。

这句话中的 It 是形式主语，真正主语是 that 引导的从句，翻译时增加了泛指的“人们”来作主语。

例句 11：

In most wire lines, two conductors, a go and a return, are used for signal transmission. 在大多数有线线路中，用两根导线：一根去线和一根回线来传输信号。

本句中的 go 和 return 是抽象名词，当它们被赋予某种具体的含义时，需要补充适当的、具体的名词来表达含义。

例句 12：

In the implementation phase, you create the actual programs. 在实现阶段，建立真正的程序。

本句中的人称代词 you 是泛指，常常不译。

例句 13：

A database system is divided into several modules to achieve the overall functionality. 数据库系统通过分成几个模块来完成总体功能。

这里的不定冠词 a 表示一类，而非数量一，不译出来，译文更简洁，更顺畅。

例句 14：

In the design phase, the systems are determined, and the design of the files and/or the databases is completed. 设计阶段需要确定系统，完成文件和/或数据库的设计。

英语求形合，并列关系通常加上 and，汉语求意合，句中的逻辑关系往往不言自明。因此，在本句中，the systems are determined 和 the design of the files and/or the databases is completed 两个并列的分句，翻译时去掉了 and，使译文更符合汉语的习惯。