Starting a Software Project

软件项目启动

Part 1

Listening & Speaking

Unit 1

Dialogue: Starting a Software Project

(Kevin, Sharon, and Jason are three ***sophomores*** *in the school 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.

[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?

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.

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** progress 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 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 requirements analysis based on the original requirements, and complete a formal Software Requirements **Specification** as our guidance of design.

[2] Replace with: But I am afraid that I have not enough business knowledge about hotel management.

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 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.

Exercises

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

Words

sophomore['sɔfəmɔ:(r)] *n.* 大学二年级学生
rough[rʌf] *adj.* 初步的,粗略的
specification[ˌspesifi'keiʃn] *n.* 说明书,规范
reminder[ri'maində(r)] *n.* 提醒,提示

Phrases

in charge of 负责,领导
a suite of 一系列,一套
take charge of 担任,监管
talk over 商议,讨论

Abbreviations

UI User Interface 用户界面

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 15 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
practice['præktis] *n.* 实践,通常的做法,惯例
assimilate[ə'siməleit] *v.* 透彻理解,消化
primarily[prai'merəli] *adv.* 原来,根本上
exhaustive[ig'zɔ:stiv] *adj.* 详尽的,彻底的,全面的
address[ə'dres] *v.* 处理,满足,论述,重点提出
law[lɔ:] *n.* 规则,法则
framework['freimwə:k] *n.* 构架,体系结构,准则
prototype['prəutətaip] *n.* 原型
discipline['disəplin] *n.* 学科,方法

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 numbered spaces with the appropriate 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 40 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 author's prediction seems safe. "No Silver Bullet" has ___20___ more and more **spirited** discussion in the **literature** than has The Mythical Man-Month.

Words

mythical['miθikl] *adj.* 神话的，虚构的
peculiarly[pi'kju:liəli] *adv.* 特有地，特别地
epsilon['epsilɔn; ep'sailən] *n.* 小（或近于零）的正数
provocative[prə'vɔkətiv] *n.* 引起争论（议论，兴趣等）的
spirited['spiritid] *adj.* 热烈的
literature['litrətʃə(r)] *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. 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. 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 (Figure 1-1). 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,

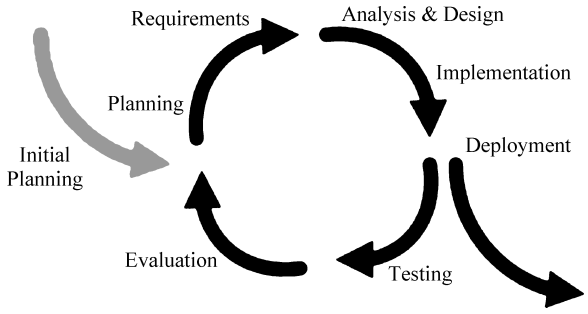

Figure 1-1 Iterative Model of Software Engineering

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 this century will be even greater.

Words

virtually['vɜ:tʃuəli] *adv.* 事实上,实质上
utility[ju:'tiləti] *n.* 公用事业,公共事业设备
distribution[distri'bju:ʃn] *n.* 经销,分销
cost-effective 有成本效益的,划算的
notion['nəuʃn] *n.* 概念,观念,看法
hitherto[ˌhiðə'tu:] *adv.* 迄今,至今
proposition[ˌprɔpə'ziʃən] *n.* 主张,提议,建议
major['meidʒə] *adj.* 较大的,较重要的
tumble['tʌmbl] *v.* 使倒下,搅乱
convergence[kən'vɜ:dʒəns] *n.* 一体化,集中,收敛
markedly['mɑ:kidli] *adv.* 显著地,明显地
notation[nəu'teiʃn] *n.* 符号
underlie[ˌʌndə'lai] *v.* 构成……的基础,位于……之下
essence['esns] *n.* 本质,实质
rightly['raitli] *adv.* 确实地

Phrases

result from 由……引起
integrated circuits 集成电路
place on 寄托,把……放在……上
over budget 超过预算

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 according to the text.

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. Fill in the numbered spaces with the words or phrases chosen from the box. Change the forms where necessary.

developer	connect	behind	need	involve
collaborate	cycle	begin	use	refer

Software Engineering vs. Software Development

The difference between software engineering and software development ___1___ with job function. A software engineer may be ___2___ with software development, but few software ___3___ are engineers.

To explain, software engineering ___4___ to the application of engineering principles to create software. Software engineers participate in the software development life cycle through ___5___ the client's needs with applicable technology solutions. Thus, they systematically develop processes to provide specific functions. In the end, software engineering means ___6___ engineering concepts to develop software.

On the other hand, software developers are the driving creative force ___7___ programs. Software developers are responsible for the entire development process. They are the ones who ___8___ with the client to create a theoretical design. They then have computer programmers create the code ___9___ to run the software properly. Computer programmers will test and fix problems together with software developers. Software developers provide project leadership and technical guidance along every stage of the software development life ___10___.

Ⅳ. Translate the following passages into Chinese.

Software Evolution

In software engineering, software evolution is referred to as the process of developing, maintaining and updating software for various reasons. Software changes are inevitable because there are many factors that change during the life cycle of a piece of software. Some of these factors include:

- Requirement changes
- Environment changes
- Errors or security breaches
- New equipment added or removed
- Improvements to the system

For many companies, one of their largest investments in their business is for software and software development. Software is considered a very critical asset and management wants to ensure they employ a team of software engineers who are devoted to ensuring that the software system stays up-to-date with ever evolving changes.

Section B: 5 Areas of Software Engineering AI will Transform

The 5 major **spheres** of software development—software design, software testing, GUI testing, strategic decision making, and automated code generation—are all areas where AI can help. A majority of interest in applying AI to software development is already seen in automated testing and bug detection tools. **Next in line** are the software design **precepts**, decision-making strategies, and finally automating software **deployment pipelines**.

Let's take an **in-depth** look into the areas of high and medium interest of software engineering impacted by AI according to the Forrester Research report (Figure 1-2).

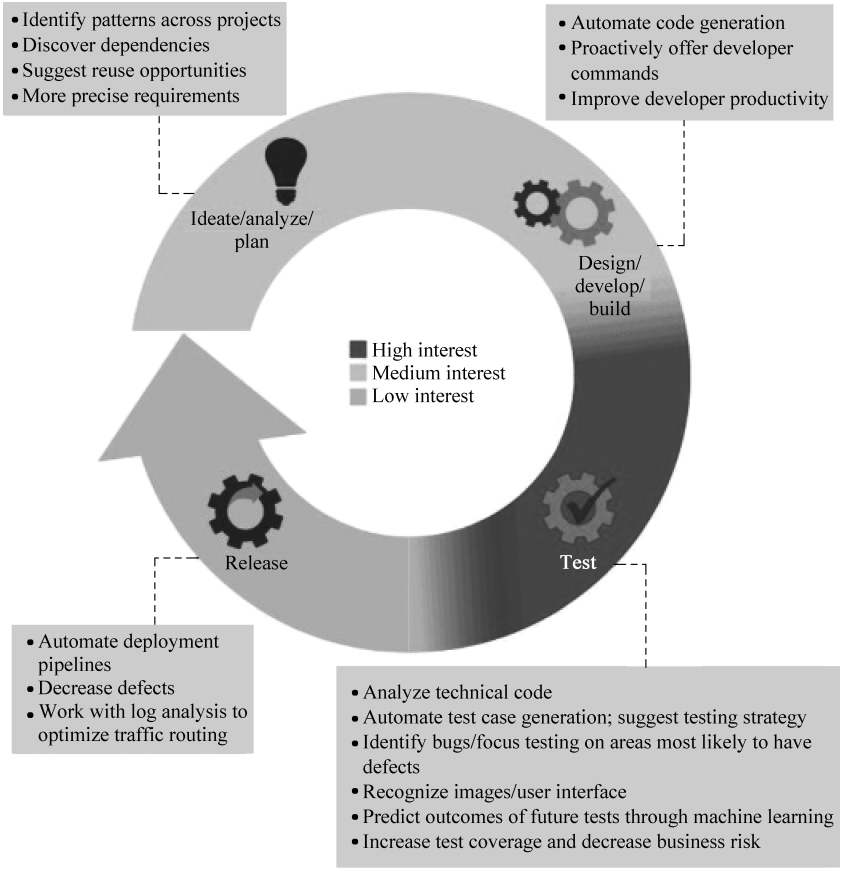

Figure 1-2 The Areas of High and Medium Interest of Software Engineering Impacted by AI According to the Forrester Research Report

1. Software design

In software engineering, planning a project and designing it **from scratch** need designers to apply their specialized learning and experience to come up with alternative solutions before **settling on** a definite solution.

A designer begins with a **vision** of the solution, and after that **retracts** and forwards

investigating plan changes until they reach the desired solution. Settling on the correct plan choices for each stage is a tedious and mistake-prone action for designers.

Along this line, a few AI developments have demonstrated the advantages of enhancing traditional methods with intelligent specialists. The **catch** here is that the operator behaves like an individual partner to the client. This **associate** should have the capacity to offer **opportune** direction on the most **proficient** method to do design projects.

For instance, take the example of AIDA—the Artificial Intelligence Design Assistant, deployed by Bookmark (a website building platform). Using AI, AIDA understands a user's needs and **desires** and uses this knowledge to create an appropriate website for the user. It makes selections from millions of combinations to create a website style, focus, image and more that are customized for the user. In about 2 minutes, AIDA designs the first version of the website, and from that point it becomes a drag and drop operation.

2. Software testing

Applications interact with each other through countless **APIs**. They **leverage** legacy systems and grow in complexity every day. Increase in complexity also leads to its fair share of challenges that can be overcome by machine-based intelligence. AI tools can be used to create test information, explore information authenticity, advancement and examination of the scope and also for test management.

Artificial intelligence, trained **right**, can ensure the testing performed is error free. Testers **freed from** repetitive manual tests thus have more time to create new automated software tests with sophisticated features. Also, if software tests are repeated every time source code is modified, repeating those tests can be not only time-consuming but extremely costly. AI comes to the rescue once again by automating the testing for you!

With AI automated testing, one can increase the overall scope of tests leading to an overall improvement of software quality.

Take, for instance, the Functionize tool. It enables users to test fast and release faster with AI enabled cloud testing. The users just have to type a test plan in English and it will be automatically get converted into a functional test case. The tool allows one to elastically scale functional, load, and performance tests across every browser and device in the cloud. It also includes **self-healing** tests that update autonomously in real-time.

SapFix is another AI Hybrid tool deployed by Facebook which can automatically generate **fixes** for specific bugs identified by "Sapienz" [1]. It then proposes these fixes to engineers for approval and deployment to production.

3. GUI testing

Graphical User Interfaces (GUI) has become important in interacting with today's

software. They are increasingly being used in critical systems and testing them is necessary to **avert** failures. With very few tools and techniques available to aid in the testing process, testing GUIs is difficult.

Currently used GUI testing methods are **ad hoc**. They require the test designer to perform **humongous** tasks like manually developing test cases, identifying the conditions to check during test execution, determining when to check these conditions, and finally evaluate whether the GUI software is adequately tested. **Phew**! Now that is a lot of work.

Also, not forgetting that if the GUI is modified after being tested, the test designer must change the test suite and perform re-testing. As a result, GUI testing today is resource intensive and it is difficult to determine if the testing is adequate.

Applitools is a GUI tester tool **empowered** by AI. The Applitools Eyes **SDK** automatically tests whether visual code is functioning properly or not. Applitools enables users to test their visual code just as thoroughly as their functional UI code to ensure that the visual look of the application is as you expect it to be. Users can test how their application looks in multiple screen layouts to ensure that they all fit the design.

It allows users to keep track of both the webpage behavior, as well as the look of the webpage. Users can test everything they develop from the functional behavior of their application to its visual look.

4. Using Artificial Intelligence in Strategic Decision-Making

Normally, developers have to **go through** a long process to decide what features to include in a product. However, machine learning AI solution trained on business factors and past development projects can analyze the performance of existing applications and help both teams of engineers and business **stakeholders** like project managers to find solutions to maximize impact and cut risk.

Normally, the transformation of business requirements into technology specifications requires a significant timeline for planning. Machine learning can help software development companies to speed up the process, deliver the product in lesser time, and increase revenue within a short span.

AI Canvas is a well-known tool for strategic decision-making. The Canvas helps identify the key questions and feasibility challenges associated with building and deploying machine learning models in the enterprise.

The AI Canvas is a simple tool that helps enterprises organize what they need to know into seven categories, **namely**—Prediction, Judgment, Action, Outcome, Input, Training and feedback. Clarifying these seven factors for each critical decision throughout the organization will help in identifying opportunities for AIs to either reduce costs or enhance performance.

5. Automatic Code Generation/Intelligent Programming Assistants

Coding a huge project from scratch is often **labor intensive** and time consuming. An intelligent AI programming assistant will reduce the workload by a great extent.

To combat the issues of time and money constraints, researchers have tried to build systems that can write code before, but the problem is that these methods aren't that good with ambiguity. Hence, a lot of details are needed about what the target program aims at doing, and writing down these details can be as much work as just writing the code. With AI, the story can be **flipped**.

"Bayou"—an AI—based application is an intelligent programming assistant. It began as an initiative aimed at extracting knowledge from online source code **repositories** like GitHub. Bayou follows a method called neural sketch learning. It trains an artificial neural network to recognize high-level patterns in hundreds of thousands of Java programs. It does this by creating a "sketch" for each program it reads and then associates this sketch with the "intent" that lies behind the program. This **DARPA** initiative aims at making programming easier and less error prone.

6. Summing it all up

Software engineering has seen massive transformation over the past few years. AI and software intelligence tools aim to make software development easier and more reliable. According to a Forrester Research report on AI's impact on software development, automated testing and bug detection tools use AI the most to improve software development.

It will be interesting to see the future developments in software engineering empowered with AI. It's expected to have faster, more efficient, more effective, and less costly software development cycles while engineers and other development personnel focus on **bettering** their skills to make advanced use of AI in their processes.

Words

sphere[sfiə(r)] *n.* 范围
precept['pri:sept] *n.* 规则
in-depth 深入详尽的,彻底的
vision['viʒn] *n.* 愿景,视野,想象
retract[ri'trækt] *v.* 撤回,收回(协议、承诺等)
catch[kætʃ] *n.* 隐藏的困难,暗藏的不利因素
associate[ə'səusieit, ə'səusiət] *n.* 同事,伙伴
opportune['ɔpətju:n] *adj.* 恰好的,适当的
proficient[prə'fiʃnt] *adj.* 娴熟的,精通的,训练有素的
desire[di'zaiə(r)] *n.* & *v.* 愿望,欲望,渴望

leverage['li:vəridʒ] *v.* 利用,施加影响
right[rait] *adv.* 正确地,恰当地,彻底地
self-healing 自愈
fix[fiks] *n.* 仓促的解决办法
avert[ə'vɜ:t] *v.* 避免,防止
humongous[hju:'mʌŋgəs] *adj.* 巨大无比的,极大的
phew[fju:] *int.* 唉呀,唷(表示不快、惊讶的声音)
empower[im'pauə(r)] *v.* 使能够,授权
stakeholder['steikhəuldə(r)] *n.* 利益相关者,干系者
namely['neimli] *adv.* 即,也就是
flip[flip] *v.* 翻转,快速翻动
repository[ri'pɔzətri] *n.* 知识库,仓库,存储库
better['betə(r)] *v.* 改善,胜过,超过

Phrases

next in line (按顺序的)下一个
deployment pipeline 部署流水线
from scratch 从头做起
settle on 选定,决定
free from 免于,使摆脱
ad hoc 特别的,专门的
go through 经受,仔细检查
labor intensive 劳动密集型,人工密集
sum up 总结,概述

Abbreviations

API Application Programming Interface 应用程序接口
SDK Software Development Kit 软件开发工具包
DARPA Defense Advanced Research Projects Agency 美国国防部高级研究计划署

Notes

[1] Sapienz 是 Facebook 的 Crash 动态扫描工具。

Exercises

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

____1. "Bayou" is an intelligent requirement assistant.
____2. Coding a huge project from scratch is often very easy.

____3. The test designer must change the test suite and perform re-testing if the GUI is modified after being tested.

____4. Applitools is another AI Hybrid tool deployed by Facebook.

____5. AI Canvas is a well-known tool for programming.

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

1. How many areas of software engineering will AI transform?

(A) One

(B) Three

(C) Five

(D) Seven

2. Which of the following statements is an intelligent programming assistant?

(A) SapFix

(B) Functionize

(C) Applitools

(D) Bayou

3. Which of the following description is right?

(A) The test designer must change the test suite and perform re-testing if the GUI is modified after being tested.

(B) Bayou follows a method called neural sketch learning.

(C) Applitools is a GUI tester tool empowered by AI.

(D) All of the above.

Ⅲ. Fill in the numbered spaces with the words or phrases chosen from the box. Change the forms where necessary.

aim	leader	core	learn	evolve
out	go	corner	create	level

Transformation to Modern Software Engineering

"Transformation"—a thorough or dramatic change in form or appearance—has been ___1___ on as long as society itself. AI, blockchain and other innovations of today are, at a basic ___2___, just modern iterations of the stone, copper and iron tools our forebears put to work millennia ago. In software engineering we're very much in the throes of our own transformation. For businesses ___3___ to grow and prosper in the digital economy, the stakes are similar: get it right, adapt or die.

Accelerating technology advancements over the past two decades ___4___ an environment of continuous disruption. Business ___5___ at big, established enterprises can't afford complacency. Nimble, digital upstarts are lurking right around the ___6___, and many large companies struggle to keep pace.

Robust software is at the ___7___ of the agile, digital business, which means we have an existential imperative to figure this ___8___ and tackle several pressing questions: How is software engineering ___9___ toward "modern" software engineering? What have we ___10___ from others' attempts at transformation? What are the major challenges and risks?

Ⅳ. Translate the following passage into Chinese.

AI-based Approaches for the Management of Complex Software Projects

Artificial Intelligence (AI) has a fundamental influence on all areas of economy, administration and society. An unexpected application of AI lies in software engineering: for the first time, AI provides robust approaches for software development in order to analyze and evaluate complex software and its development processes. Repository Mining, Machine Learning, Big Data Analytics and Software Visualization enable targeted insights and powerful predictions for software quality, software development and software project management.

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 no 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