Homework1

梁钰波

16341012

1. 软件工程的定义

IEEE：软件工程是（1）将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件；（2）在（1）中所述方法的研究。

BarryBoehm：运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。

FritzBauer：在NATO会议上给出的定义：建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。

1. 解释导致 software crisis 本质原因、表现，述说克服软件危机的方法

软件危机（software crisis）指在规定的时间内编写有用而高效的计算机程序的困难性。它出现的本质原因是由于计算机能力的迅速提高和无法解决的问题的复杂性造成的。随着软件复杂性的增加，由于现有方法的不足，出现了许多软件问题。1972年图灵奖获得者 Edsger Dijkstra曾说过，软件危机的本质原因是"机器已经变得强大了几个数量级!坦率地说:只要没有机器，编程根本就不是问题;当我们有一些弱的计算机时，编程变成了一个温和的问题，现在我们有了巨大的计算机，编程也变成了一个同样巨大的问题。"通俗点说，没有计算机时，根本没有编程上的问题；有了一些没那么强的计算机时，编程的问题出现了，但问题不大；现在计算机越来越强大，编程也就越来越复杂，也随之成为一个庞大而复杂的问题。计算能力已经超过了程序员有效利用这些能力的能力。在过去的几十年中，开发了各种过程和方法来改进软件质量管理，如过程编程和面向对象编程。然而，大型的、复杂的、没有详细说明的、涉及不熟悉方面的软件项目仍然容易受到大型的、未预料到的问题的影响。

软件危机的表现：

项目运行超过预算

项目运行超过时间

软件非常低效

软件质量很差

软件常常不能满足需求

项目不可管理，代码难以维护

软件从未交付

软件危机的解决方法：

针对软件危机的本质原因，我们需要构建软件生产的最佳实践与知识体系，发展更好的软件开发和管理方法，比如面向过程编程(PP)、面向对象编程(OOP)、面向切面编程(AOP)等都是过去十年来提出的用来改善软件质量管理的方法。但软件由于其自身的复杂性，很容易被这样那样的不可预测的问题所侵袭，所以只能说尽可能地避免或者减少其问题，而很难完全解决软件危机。

软件工程诞生于60年代末期，它作为一个新兴的工程学科，主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本 、改进软件产品质量、提高软件生产率水平的目标。软件工程学从硬件工程和其他人类工程中吸收了许多成功的经验，明确提出了软件生命周期的模型，发展了许多软件开发与维护阶段适用的技术和方法，并应用于软件工程实践，取得良好的效果。

在软件开发过程中人们开始研制和使用软件工具，用以辅助进行软件项目管理与技术生产，人们还将软件生命周期各阶段使用的软件工具有机地集合成为一个整体，形成能够连续支持软件开发与维护全过程的集成化软件支援环境，以期从管理和技术两方面解决软件危机问题。此外，人工智能与软件工程的结合成为80年代末期活跃的研究领域。基于程序变换、自动生成和可重用软件等软件新技术研究也已取得一定的进展，把程序设计自动化的进程向前推进一步。在软件工程理论的指导下，发达国家已经建立起较为完备的软件工业化生产体系，形成了强大的软件生产能力 。软件标准化与可重用性得到了工业界的高度重视，在避免重用劳动，缓解软件危机方面起到了重要作用。

1. 软件生命周期

在软件工程中，软件开发过程是指在时间维度上将软件开发工作划分为不同阶段以改进设计、产品管理和项目管理的过程。它也被称为软件开发生命周期。该方法可能包括项目团队为开发或维护应用程序而创建和完成的特定交付物和工件的预先定义。大多数现代开发过程可以模糊地描述为敏捷。其他方法包括瀑布开发、原型开发、迭代和增量开发、螺旋开发、快速应用程序开发和极限编程。有些人认为生命周期“模型”是一组方法的更一般的术语，而软件开发“过程”是指由特定组织选择的特定过程的更具体的术语。例如，有许多特定的软件开发过程符合螺旋生命周期模型。该字段通常被认为是系统开发生命周期的一个子集。

1. SWEBoK 的 15 个知识域（An Overview of the SWEBOK Guide 请中文翻译其名称与简短说明）

Software Requirements：软件需求

软件需求知识域涉及软件需求的引出、协商、分析、规范和验证。在软件行业中，人们普遍认为，当这些活动执行得不好时，软件工程项目是非常脆弱的。软件需求表达了对软件产品的需求和约束，这些需求和约束有助于解决一些实际问题。

Software Design：软件设计

定义系统或组件的体系结构，组件，接口和其他特征的过程以及该过程的结果。软件设计知识域包括设计过程和最终产品。软件设计过程是软件工程生命周期活动，在该活动中，软件需求被分析，以产生对软件内部结构及其行为的描述，这些描述将作为软件构建的基础。

Software Construction：软件构建

指通过结合详细设计，编码，单元测试，集成测试，调试和验证来详细创建工作软件。软件构建知识域包括与满足其需求和设计约束的软件程序开发相关的主题。这个知识域涵盖了软件构建基础;管理软件建设;施工技术;实际问题;以及软件构建工具。

Software Testing：软件测试

测试是一项旨在通过识别缺陷来评估产品质量并改进产品质量的活动。软件测试涉及到在一组有限的测试用例上根据预期行为动态地验证程序的行为。

Software Maintenance：软件维护

软件维护包括增强现有功能，调整软件以在新的和修改的操作环境中运行，以及纠正缺陷。这些类别称为完善，自适应和纠正性软件维护。软件维护知识域包括软件维护的基本知识(维护的性质和需要、维护的类别、维护费用);软件维护中的关键问题(技术问题、管理问题、维护成本估算、软件维护度量);维护过程;软件维护技术(程序理解、重组工程、逆向工程、重构、软件退役);灾难恢复技术和软件维护工具。

Software Configuration Management：软件配置管理

软件配置管理是在不同的时间点识别系统配置的规程，以便系统地控制配置的更改，并在整个软件生命周期中维护配置的完整性和可追溯性。软件配置管理知识域包括单片机过程的管理;软件配置识别、控制、状态核算、审计;软件发布管理和交付;以及软件配置管理工具。

Software Engineering Management：软件工程管理

软件工程管理涉及规划，协调，测量，报告和控制项目或程序，以确保软件的开发和维护是系统化的，规范化的和量化的。

Software Engineering Process：软件工程过程

软件工程涉及软件生命周期过程的定义、实现、评估、度量、管理和改进。所涵盖的主题包括过程实现和变更(过程基础结构、过程实现和变更的模型以及软件过程管理);过程定义(软件生命周期模型和过程，过程定义、过程适应和过程自动化的符号);过程评估模型和方法;测量(过程测量、产品测量、测量技术、测量结果质量);以及软件过程工具。

Software Engineering Models and Methods：软件工程模型和方法

软件工程模型和方法知识域解决了包含多个生命周期阶段的方法;针对特定生命周期阶段的方法由其他KA覆盖。所涵盖的主题包括建模(软件工程模型的原理和属性;语法、语义、不变量;(前置条件、后置条件和不变量);模型的类型(信息、结构和行为模型);分析(对正确性、完整性、一致性、质量和交互进行分析;可追溯性;和权衡分析);以及软件开发方法(启发式方法、正式方法、原型方法和敏捷方法)。

Software Quality：软件质量

软件质量是一个普遍存在的软件生命周期问题，在许多SWEBOK V3 知识域中都得到了解决。此外，软件质量知识域还包括软件质量的基础(软件工程文化、软件质量特征、软件质量的价值和成本、软件质量改进);软件质量管理过程(软件质量保证、验证和验证、评审和审计);以及实际的考虑(缺陷特性、软件质量度量和软件质量工具)。

Software Engineering Professional Practice：软件工程专业实践

软件工程专业实践是指软件工程师必须具备的知识、技能和态度，以一种专业、负责和道德的方式来实践软件工程。软件工程专业实践知识域涵盖专业(专业行为、专业协会、软件工程标准、雇佣合同、法律问题);伦理准则;团队动态(在团队中工作，认知问题的复杂性，与利益相关者的互动，处理不确定性和模糊性，处理多元文化环境);和沟通能力。

Software Engineering Economics ：软件工程经济学

软件工程经济学关注的是在业务环境中做出决策，以使技术决策与组织的业务目标保持一致。所涵盖的主题包括软件工程经济学的基本原理(建议、现金流量、金钱的时间价值、规划期限、通货膨胀、折旧、重置和退休决定);非营利性决策(成本效益分析、优化分析);评估、经济风险和不确定性(评估技术、风险和不确定性下的决策);以及多属性决策(值和度量尺度、补偿和非补偿技术)。

Computing Foundations：计算基础

计算基础KA涵盖了提供软件工程实践所需的计算背景的基础主题。主题包括问题解决技术、抽象、算法和复杂性、编程基础、并行和分布式计算的基础、计算机组织、操作系统和网络通信。

Mathematical Foundations：数学基础

数学基础KA涵盖了提供软件工程实践所必需的数学背景的基础主题。涵盖的主题包括集合，关系和功能;基本命题和谓词逻辑;证明技巧;图论和树;离散概率;语法和有限状态机;数论。

Engineering Foundations：工程基础

工程基础KA涵盖了提供软件工程实践所必需的工程背景的基础主题。涵盖的主题包括经验方法和实验技术;统计分析;测量和指标;工程设计;仿真与建模;根本原因分析。

1. 简单解释 CMMI 的五个级别。例如：Level 1 - Initial：无序，自发生产模式。

Level1-initial: 无序，自发生产模式、不可预测的、控制不良、反应性差、被动的

Level2-managed: 有基本的管理过程，以项目为特征的过程，通常是反应性的和被动的

Level3-defined: 以组织和主动性为特征的过程(项目根据组织的标准定制其过程)

Level4-Quantitatively Managed: 测量和控制的过程，高质量管理、标准化、量化管理、可控

Level5-Optimizing: 关注过程改进，不断对管理过程进行优化和改善

1. 用自己语言简述 SWEBok 或 CMMI （约200字）

CMMI（Capability Maturity Model Integration）即能力成熟度模型。CMMI是一套融合多学科的、可扩充的产品集合，是一个过程级别的改进训练和评估程序，本质上是软件管理工程的一个部分，目的是帮助软件企业对软件工程过程进行管理和改进，从而能按时地、不超预算地开发出高质量的软件。它由ISACA的子公司CMMI研究所管理，由卡内基梅隆大学开发。许多美国国防部和美国政府合同，尤其是在软件开发方面，都要求这样做。CMMI为流程定义了以下成熟度级别：初始、管理、定义、定量管理和优化。CMMI模型为开发或改进流程提供指导，以满足组织的业务目标，也可以用作评估组织过程成熟度的框架，通过使用知识（能力）的不同，用不同级别表示开发企业的成熟性，不仅给出了企业能力提升的路径，也给出了企业能力评估的事实标准。