COMP130015.02
软件工程

1. 软件工程概述

复旦大学计算机科学技术学院
沈立炜
shenliwei@fudan.edu.cn

什么是软件

- 广义的“软件”
  - ✓ 任何构筑在建筑、机械、电子等物理设施和设备之上可灵活调整的流程、规则、服务和各种处理逻辑
  - ✓ 例如，对于一个商贸市场而言：整个市场的房屋建筑、摊位及其他相关设施和设备属于硬件；市场为商户和顾客提供的各种管理流程和附加服务则属于软件
- 计算机软件
  - ✓ 计算机系统中与硬件相对的部分
  - ✓ 在通用计算机硬件之上面向特定应用目标实现的解决方案
  - ✓ 根本特性
    - 能够在通用计算机硬件之上运行
    - 能够灵活面向不同的应用目标实现相应的解决方案
    - 内容上包括程序及其文档以及相关的数据

软件发展历史：1940年代-1960年代

- 软硬件一体化：软件以依附于电子计算机上的完成特定任务的程序的形式出现
- 计算机应用局限在国防军工以及科学计算等少数的领域之中
- 在此期间，开发人员所使用的编程语言从机器语言、汇编语言逐渐发展到高级语言，完成编程任务的效率逐渐提高

软件发展历史：1960年代-2000年代

- 标志性事件：1960年代IBM宣布软件与硬件解除绑定并对软件单独计价
- 此后整个计算机和信息产业进入一个全新的发展时期，软件产业蓬勃发展
- 软件产品主要以软盘和光盘等物理介质作为分发和销售的载体
- 越来越多的设备拥有网络通信能力，网络化软件逐渐成为主流
- 涌现了一些著名的软件企业，例如
  - ✓ Microsoft（操作系统与办公软件）、Oracle（数据库）
  - ✓ ERP领域：SAP、Oracle、用友、金蝶

软件产品分类

- 按照软件应用领域
  - ✓ 通用软件产品：实现通用需求，一般通过市场向广大客户和用户直接进行销售，例如办公软件、工具软件、游戏等
  - ✓ 定制化软件产品：面向特定客户实现定制化的需求，例如面向特定客户开发的CRM、ERP、财务管理系统等
- 按照软件的部署和运行形态
  - ✓ 信息系统：运行在通用的服务器和个人电脑等通用计算机上，主要功能是信息加工和处理，例如信息填报、业务审核、流程管理、统计分析、报表打印等
  - ✓ 嵌入式系统：运行在生产设备（如机床）、家用电器（如冰箱）、特种设备（如月球车）中，通过嵌入在设备中的有限的计算和存储资源实现设备管理和控制等方面的功能

软件发展历史：2000年代-2020年代

- 基于互联网的应用软件和服务不断涌现并蓬勃发展，同时也产生了新的基于在线服务的软件分发方式和商业模式
  - ✓ 催生了各种各样的互联网服务和商业模式，包括搜索引擎、网络购物、在线社交、网络游戏、互联网金融等
  - ✓ 产生了众多著名的互联网企业，包括美国的Google、FaceBook、Twitter、Amazon以及中国的阿里巴巴、腾讯、百度、字节跳动等
- 越来越多的应用提供了移动客户端访问方式
  - ✓ 新的软件分发和销售模式：软件应用商店，例如苹果的App Store、谷歌的Google Play以及华为应用市场等
  - ✓ 围绕各种软硬件平台又形成了各种软件生态系统，例如苹果、安卓、鸿蒙等软件生态系统

软件发展历史：2020年代-

- 软件表现出了越来越强的渗透性，其应用的触角已经深入我们社会经济生活的方方面面
- 软件已成为信息化社会不可或缺的基础设施
  - ✓ 软件自身已成为信息技术应用基础设施的重要构成成分，以平台方式为各类信息技术应用和服务提供运行支撑
  - ✓ 软件正在“融入”到支撑整个人类经济社会运行的“基础设施”中，特别是随着互联网向物理世界的拓展延伸、并与其他网络的不断交汇融合
- 软件重塑了从休闲娱乐、人际交往到生产生活、国计民生等社会经济的方方面面，“软件定义一切”日益成为一种现实

《中国学科发展战略：软件科学与工程》（国家自然科学基金委员会等，2021）

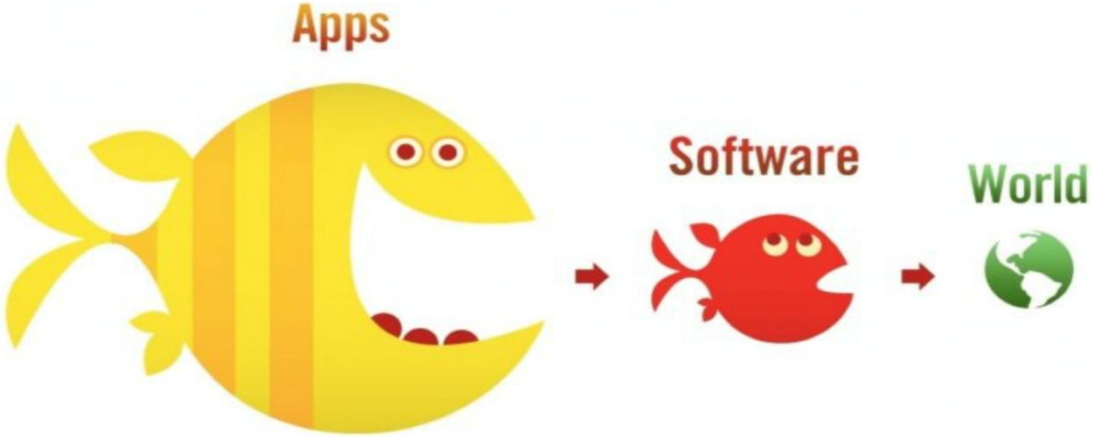
软件正在吞噬世界...

Marc Andreessen

Netscape创始人之一

Facebook、Linkedin、HP等公司董事

硅谷风投公司Andreessen-Horowitz合伙人



In the future every company will become a software company.

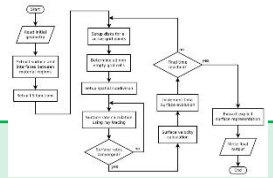
Marc Andreessen. Why Software Is Eating the World. Wall Street Journal, 2011.

课堂讨论：软件改变世界

课堂讨论：软件及相关技术的发展改变了哪些行业和领域？结合自身的体会和理解举例说明

软件开发的一般过程

抽象问题描述



问题理解

编写程序

程序

```
double dMax(double *dArray, int iLenOfArray)
{
    double max = dArray[0];
    int iCount;

    for (iCount = 0; iCount < iLenOfArray; ++iCount)
    {
        if (dArray[iCount] > max)
        {
            max = dArray[iCount];
        }
    }

    return max;
}
```

问题抽象

开发人员（Developer）

部署运行

现实世界问题，如计算工资

现实世界

计算设备，如服务器、PC机

计算机世界

软件开发是一个从问题空间到解空间的知识转换过程

如何理解软件工程

- 软件工程可以理解为软件开发的工程化或工程化的软件开发
- 基本追求：质量、成本、效率
- 此前大家所熟悉的“编程”与工程化软件开发的区别
  - ✓ 所编写的程序没有针对来自现实世界的需求
  - ✓ 不考虑相关的质量要求
  - ✓ 没有按照工程化的过程进行开发
  - ✓ 不考虑长期演化和维护的问题

“工程”的基本内涵

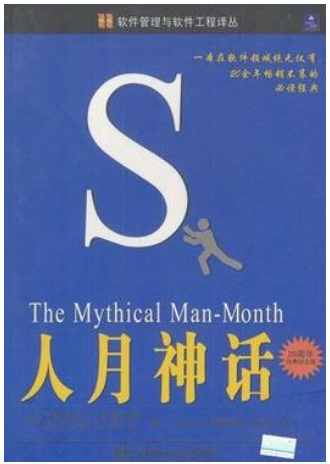
- 与桥梁工程、汽车工程、船舶工程等其他工程门类一样，软件工程也具备“工程”（Engineering）的基本内涵和特点
  - ✓ **过程标准化**：通常包含一系列明确定义的过程，包括相关的活动、所使用的工具、参与的角色以及具体工作的指南等，相关过程经常可以通过标准和规范等手段进行明确总结和描述
  - ✓ **理论和实践支撑**：所涉及的原则和方法有相应的理论支撑或者体现了长期实践经验的总结
  - ✓ **质量有保障**：项目实施过程稳定、可控，所完成的项目或所交付的产品质量在很大程度上能够得到保障，成功经验很大程度上可总结、可学习、可重复
  - ✓ **追求实用**：遵循实用性原则，强调“足够好”（good enough）而非“完美”（perfect），注重成本与效益等方面的权衡

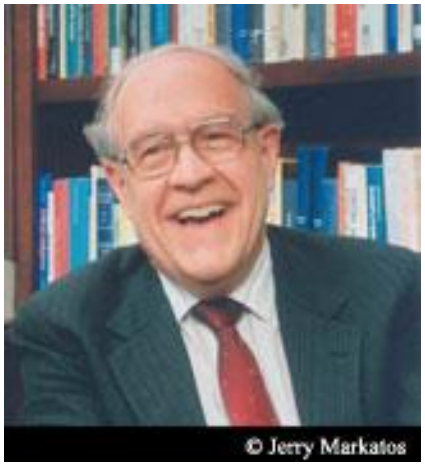
软件工程诞生的直接原因：软件危机

- 软件诞生初期开发方式类似于手工作坊，成功主要依赖于开发人员个人的能力和经验，项目完成时间以及所交付的软件质量都存在较大的不确定性
- 随后出现了所谓的软件危机（Software Crisis），表现形式包括
  - ✓ 软件开发进度难以预测
  - ✓ 软件开发成本难以控制
  - ✓ 用户的产品功能要求难以满足
  - ✓ 软件产品质量无法保证
  - ✓ 软件产品难以维护

IBM360系统的例子

- 开发时间：1963-1966年
- 投入人力：5000人年
- 代码量：100万行
- 耗资数亿美元
- 结局
  - ✓ 发布推迟，成本超支好几倍
  - ✓ 需要的内存比计划多很多
  - ✓ 第一次发布时还不能很好运行
  - ✓ 直到发布了好几次以后（每次都会修订上千个错误）





Frederick Brooks

图灵奖得主

IBM 360系统之父

Brooks的描述

.....正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难。.....程序设计工作正像这样一个泥潭，.....一批批程序员被迫在泥潭中拼命挣扎，.....谁也没有料到问题竟会陷入这样的困境.....

软件工程的诞生

1968年NATO会议上第一次提出了软件工程的概念
从此，软件工程从计算机学科体系中脱颖而出

Doug McIlroy on Software Components, 1968

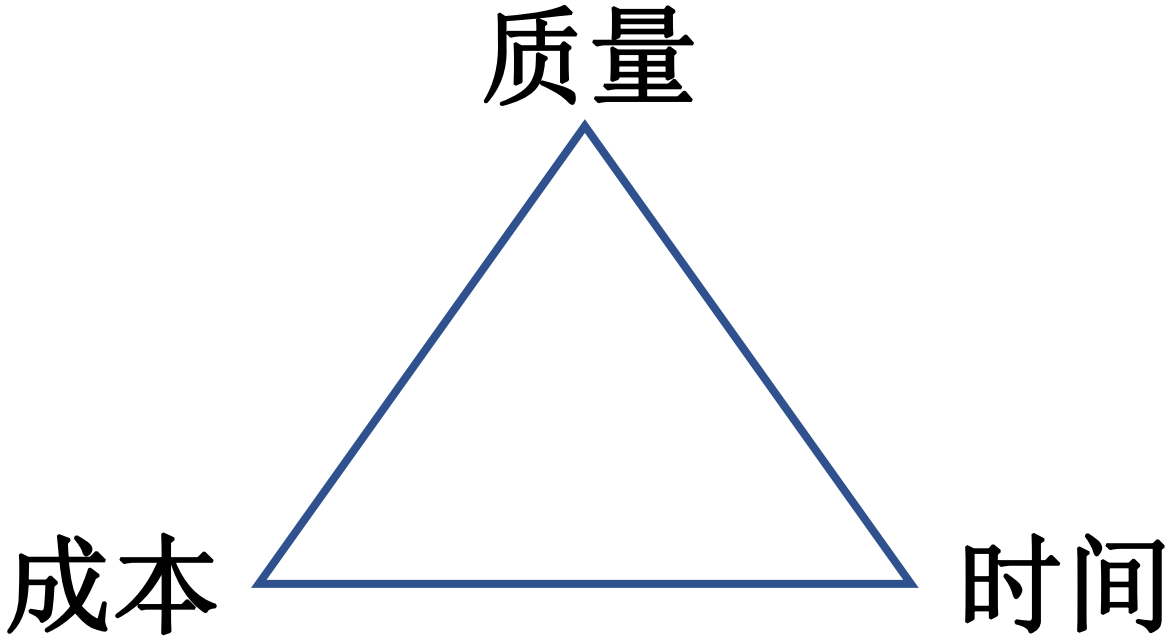
Fritz Bauer, NATO, 1968

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

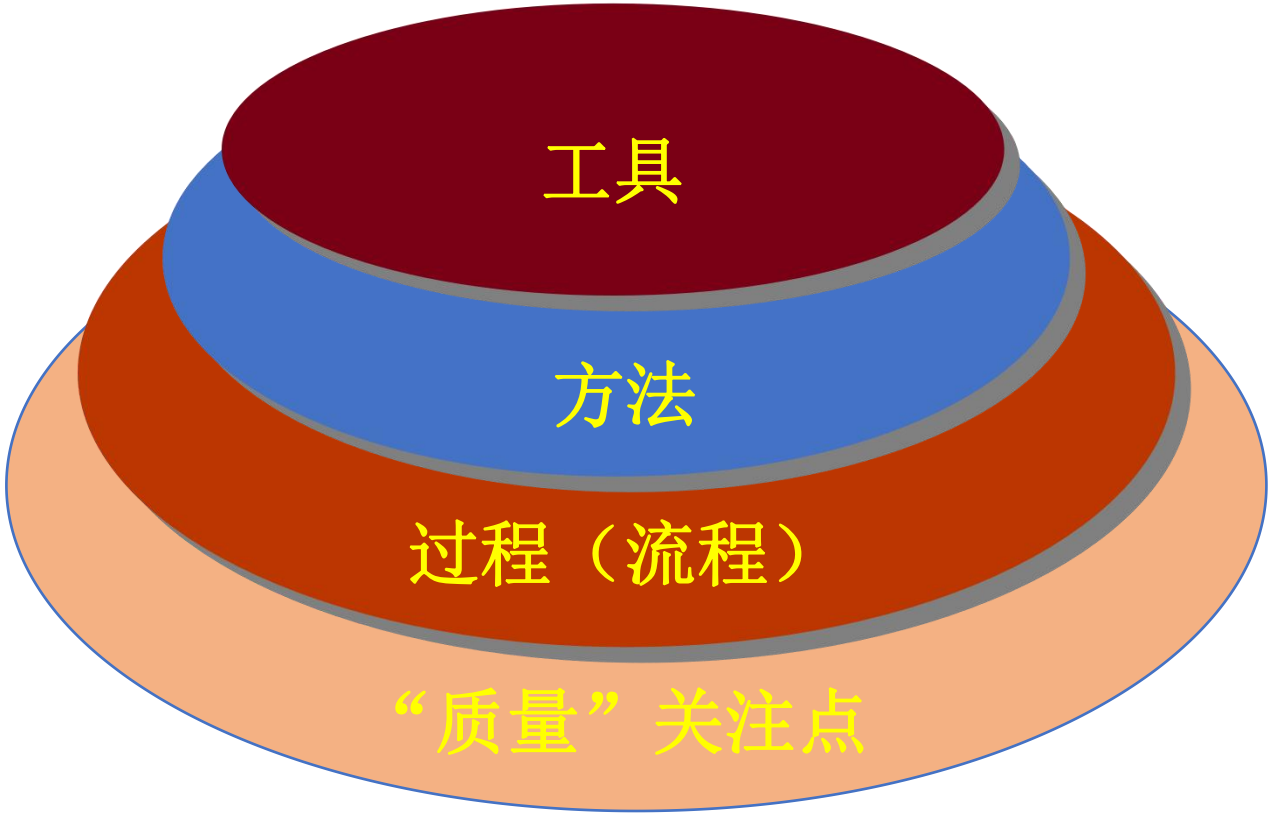
软件工程的根本特性

- **可靠性**：所交付的软件应该具有可靠性
- **运行效率**：所交付的软件能够在真实的计算设备上高效运行
- **经济性**：所采用的过程和方法需要考虑成本效益
- **工程**：实现以上目标的手段是建立并使用合理的工程原则

软件工程的基本关注点



软件工程的内容层次



软件工程的根本性困难

- 软件开发就是将问题空间(所面临的特定问题域)转化为解空间(代码、文档)的过程
- 根本性困难
  - ✓ 不可见的逻辑产品（对比：物理学和化学对于制造业、建筑业等成熟领域的工程技术支撑)
  - ✓ 问题空间和解空间之间的巨大鸿沟
  - ✓ 面临的领域纷繁复杂、分布广泛，不同软件系统之间的差异性很大
  - ✓ 软件系统的复杂度越来越高（远高于计算机硬件)

隐藏的复杂性

课堂讨论：工程化软件开发

课堂讨论：如何理解“工程化软件开发”与到目前为止大家所熟悉的“编程”之间的区别？

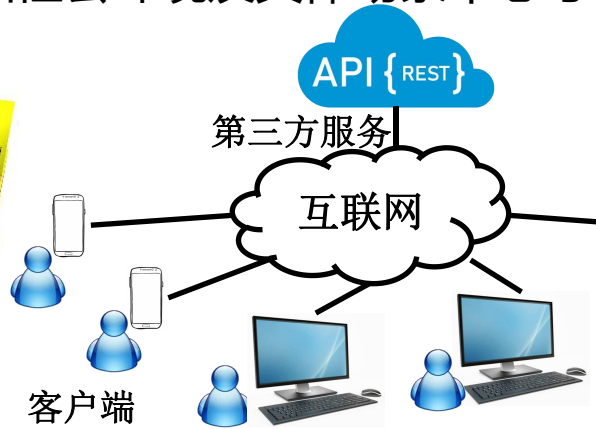
软件工程的系统观与演化观

- 从空间和时间两个方面将软件及其开发过程置于一个更大的社会、经济和技术环境之中，并从发展和变化的角度去理解
- 空间维度：系统观
  - ✓ 对软件系统整体结构及各部分关系的整体理解
  - ✓ 对于用户及其他涉众而言，真正有价值的是完整的系统而非其中的软件
- 时间维度：演化观
  - ✓ 对软件系统的持续发展变化有一个全面的理解
  - ✓ 以发展的眼光看待软件系统，考虑软件长期演化和维护的需要

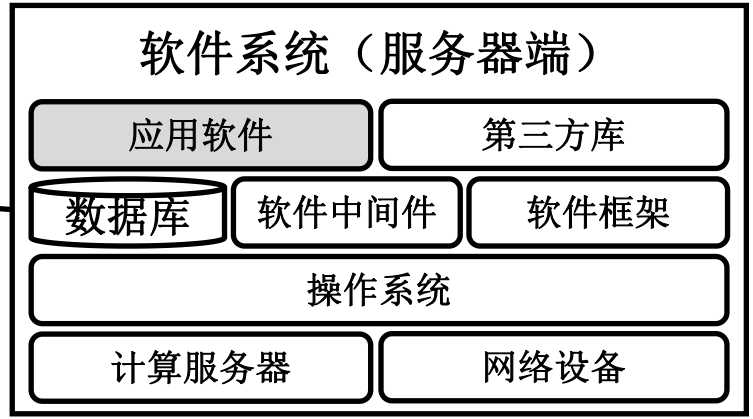
软件工程系统观的两个方面

- 软件系统的计算环境及技术栈
  - ✓ 现代软件系统的复杂性导致应用软件开发通常都需要在一套软硬件技术栈的基础上进行开发
  - ✓ 需要对由此产生的复杂依赖关系、配置要求以及相关风险有所了解
- 软件系统的物理和社会环境
  - ✓ 软件系统需要与环境中的社会和物理要素共同作用从而实现用户需求
  - ✓ 需要在物理和社会环境及具体场景中思考和理解软件的角色和需求

社会和物理环境

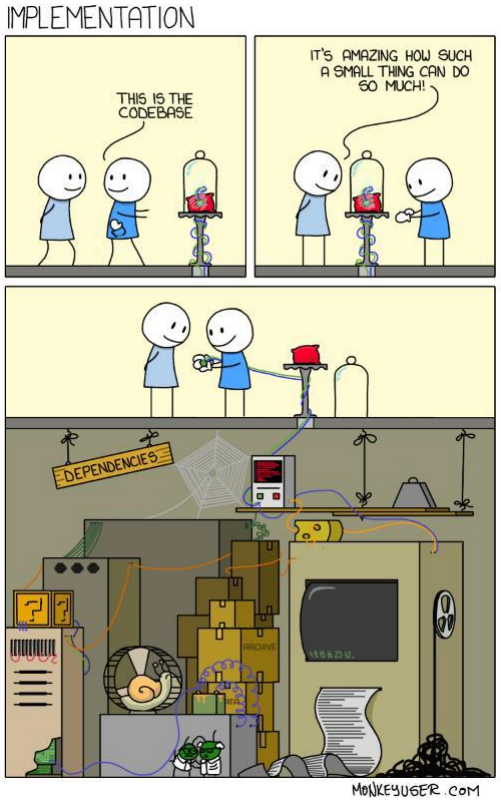


客户端



隐藏的复杂性

软件维护的冰山



隐藏的依赖

课堂讨论：软件依赖

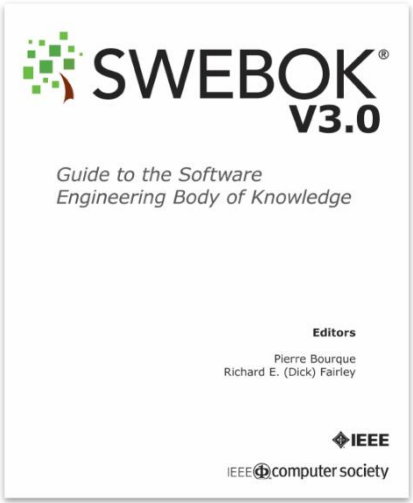
课堂讨论：根据自己的经历举例说明你写过的程序存在哪些依赖关系？这些依赖如何影响程序的正常运行？

软件工程演化观的两个方面

- 软件系统的持续演化
  - ✓ 软件在生存期内需要不断演化以适应客户与用户需求以及环境的变化，从而保持其“有用性”：缺陷修复、性能改进、环境变化适应、新技术应用等
  - ✓ 主动选择对软件的设计和实现进行重构，从而改进其可维护性和可扩展性
- 软件系统的迭代和演化式开发
  - ✓ 现代软件开发普遍采用了敏捷方法，通过很短的迭代周期持续交付可运行的软件并获得用户反馈
  - ✓ 开发运维一体化（即DevOps）进一步通过频繁的部署和持续的监控更快地获得使用反馈
  - ✓ 软件开发与演化已经融为一体，无法完全区分开

软件工程知识体系

- 任何职业都是建立在相应的知识体系基础上
  - ✓ 制定职业培训计划
  - ✓ 专家资格认证
  - ✓ 职业许可认证



软件工程知识体系

Software Engineering Body of Knowledge (SWEBOK)

SWEBOK总体结构（V3.0）

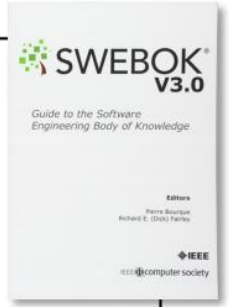
知识体系

- 知识领域 (Knowledge Area)
- 知识领域 (Knowledge Area)

Table I.1. The 15 SWEBOK KAs
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

SWEBOK V4.0内容变化

SWEBOK V3



Requirements

Design
Construction
Testing

Maintenance
Configuration Management
Engineering Management
Process
Models and Methods
Quality

Professional Practice
Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

SWEBOK V4

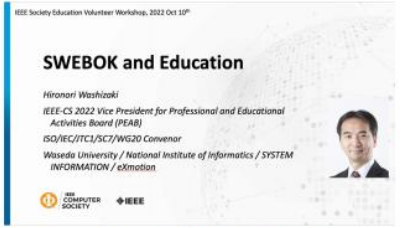
Requirements
Architecture
Design
Construction
Testing
Operations
Maintenance
Configuration Management
Engineering Management
Process
Models and Methods
Quality
Security
Professional Practice
Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

Agile testing
...

Agile, DevOps

Agile security
...

反映了现代软件工程，实践变化和更新，知识体系增长和新发展的领域



V4.0公开评审中：https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-committee/swebok-evolution

软件工程师的社会责任

国家需求
社会需要

现实世界

伦理道德
社会影响

理解

应用

软件定义一切

职业道德
工匠精神

交付

可靠可信
自主可控

软件工程师

软件系统

软件工程师的职业道德

- 软件的开发及其应用必须符合法律规定
  - ✓ 不能利用软件技术作恶（例如开发软件进行恶意攻击或其他非法活动）
  - ✓ 不能在软件开发中侵犯他人知识产权
- 软件的开发及其应用必须满足道德规范和职业精神
  - ✓ 保守商业和技术秘密
  - ✓ 尊重雇主和客户利益
  - ✓ 保护用户隐私
- 软件工程师必须有责任感
  - ✓ 有着高度的质量意识、追求卓越
  - ✓ 有社会责任感、追求社会公平与正义

ACM-IEEE软件工程职业道德与行为准则

ACM和IEEE联合推出了一个关于软件工程职业道德和职业行为的准则

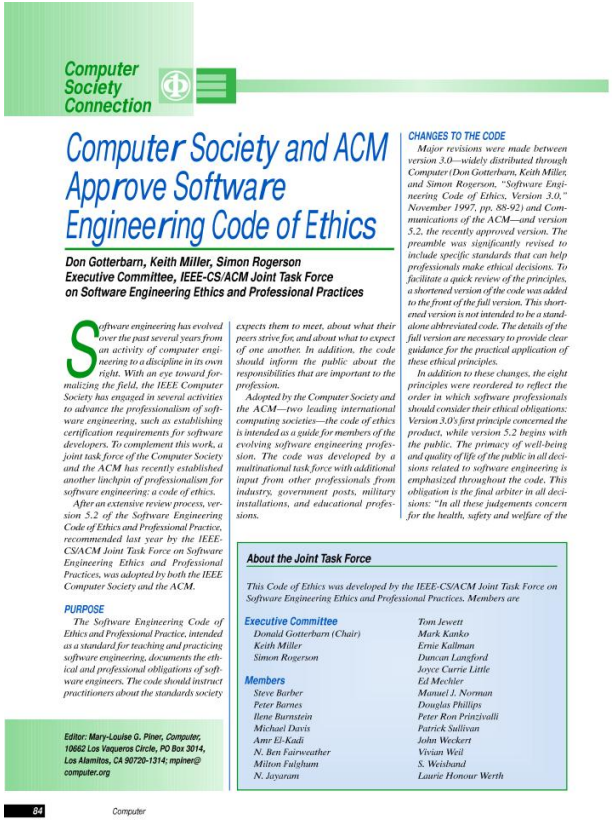
软件工程职业道德和行为准则
（ACM/IEEE-CS软件工程职业道德和行为规范联合工作组）
前言

准则的简要版对其中的愿望做了高度抽象的概括；完整版中的条款对这些愿望进行了细化，并给出了实例，用以规范软件工程专业人员的工作方式。没有这些愿望，所有的细节都会变得教条而又枯燥；而没有这些细节，愿望就会变成高调而空洞。只有将二者紧密结合才能形成有机的行为准则。

软件工程师应当作出承诺，使软件的分析、规格说明、设计、开发、测试和维护等工作对社会有益且受人尊重。基于对公众健康、安全和福利的考虑，软件工程师应当遵守以下8条原则：

1. 公众感——软件工程师应始终与公众利益保持一致；
2. 客户和雇主——软件工程师应当在与公众利益保持一致的前提下，保证客户和雇主的最大利益。
3. 产品——软件工程师应当保证他们的产品以及相关的修改尽可能满足最高的行业标准。
4. 判断力——软件工程师应当具备公正和独立的职业判断力。
5. 管理——软件工程管理者和领导者应当维护并倡导合乎道德的有关软件开发和维护的管理方法。
6. 职业感——软件工程师应当弘扬职业正义感和荣誉感，尊重社会公众利益。
7. 同事——软件工程师应当公平地对待和协助每一位同事。
8. 自己——软件工程师应当毕生学习专业知识，倡导合乎职业道德的职业活动方式。

http://www.acm.org/about/se-code



Computer Society Connection

Computer Society and ACM Approve Software Engineering Code of Ethics

Don Gotterbarn, Keith Miller, Simon Rogerson
Executive Committee, IEEE-CS/ACM Joint Task Force
on Software Engineering Ethics and Professional Practices

Software engineering has evolved over the past several years from an activity of computer engineering to a discipline in its own right. With an eye toward formalizing the field, the IEEE Computer Society has engaged in several activities to advance the professionalism of software engineering, such as establishing certification requirements for software developers. To complement this work, a joint task force of the Computer Society and the ACM has recently established another linchpin of professionalism for software engineering: a code of ethics.

After an extensive review process, version 5.2 of the Software Engineering Code of Ethics and Professional Practice, recommended last year by the IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices, was adopted by both the IEEE Computer Society and the ACM.

PURPOSE

The Software Engineering Code of Ethics and Professional Practice, intended as a standard for teaching and practicing software engineering, documents the ethical and professional obligations of software engineers. The code should instruct practitioners about the standards society expects them to meet, about what their peers strive for, and about what to expect of one another. In addition, the code should inform the public about the responsibilities that are important to the profession.

Adopted by the Computer Society and the ACM—two leading international computing societies—the code of ethics is intended as a guide for members of the evolving software engineering profession. The code was developed by a multinational task force with additional input from other professionals from industry, government posts, military installations, and educational professions.

CHANGES TO THE CODE

Major revisions were made between version 3.0—widely distributed through Computer (Don Gotterbarn, Keith Miller, and Simon Rogerson, "Software Engineering Code of Ethics, Version 3.0," November 1997, pp. 88-92) and Communications of the ACM—and version 5.2, the recently approved version. The preamble was significantly revised to include specific standards that can help professionals make ethical decisions. To facilitate a quick review of the principles, a shortened version of the code was added to the front of the full version. This shortened version is not intended to be a stand-alone abbreviated code. The details of the full version are necessary to provide clear guidance for the practical application of these ethical principles.

In addition to these changes, the eight principles were reordered to reflect the order in which software professionals should consider their ethical obligations: Version 3.0's first principle concerned the product, while version 5.2 begins with the public. The primacy of well-being and quality of life of the public in all decisions related to software engineering is emphasized throughout the code. This obligation is the final arbiter in all decisions: "In all these judgements concern for the health, safety and welfare of the

About the Joint Task Force

This Code of Ethics was developed by the IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices. Members are

Executive Committee
Donald Gotterbarn (Chair)
Keith Miller
Simon Rogerson

Members
Steve Barber
Peter Barnes
Ilene Burnstein
Michael Davis
Amr El-Kadi
N. Ben Fairweather
Milton Fulgham
N. Jayaram
Tom Jewett
Mark Kanko
Ernie Kallman
Duncan Langford
Joyce Currie Little
Ed Mechler
Manuel J. Norman
Douglas Phillips
Peter Ron Prinzivalli
Patrick Sullivan
John Weckert
Vivian Weil
S. Weisband
Laurie Honour Werth

Editor: Mary-Louise G. Piner, Computer, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; mpiner@computer.org

D. Gotterbarn, K. Miller, S. Rogerson. Computer society and ACM approve software engineering code of ethics. IEEE Computer, Vol.32, No.10, Oct.1999, pp.84-88.

课堂讨论：软件工程师的职业道德

课堂讨论：软件工程师哪些方面的行为可能涉及职业道德问题（举例说明）？

本章小结

- 软件表现出了越来越强的渗透性，其应用的触角已经深入我们社会经济生活的方方面面
- 软件应用范围的不断扩大以及软件开发要求的不断提高使得工程化的软件开发逐渐成为共识
- 理解软件工程需要建立相应的系统观和演化观
- 需要深刻理解软件工程师的社会责任

COMP130015.02
软件工程

End

1. 软件工程概述